CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

OrgTrac: Building an Android Application with Dynamic and Cost-Effective Solutions

A graduate project submitted in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering

By
Robert Dale Johnson III

December 2018

The graduate project of Robert Dale Johnson III is approved:

Kyle Dewey, Ph.D	Date
Katya Mkrtchyan, Ph.D	Date
Taehyung (George) Wang, Ph.D., Chair	Date

California State University, Northridge

Acknowledgements

It has taken many years to finally get this project complete. I want to thank everyone who has helped me throughout the lifetime and many iterations of this project. I no idea just how much time would pass prior to being able to complete this project nor did I realize just how much would be involved in completing it. I am nonetheless ecstatic that I have finally been able to complete it.

I am especially grateful for my committee members and their willingness to take on their roles in my committee with such short notice. A special thanks to Prof. Kyle Dewey for all his early feedback on my Project Proposal that helped me see the light at the end of the tunnel and for Prof. Taehyung (George) Wang for taking on the role of chair so absolutely late in the process.

As with any major effort it is my family that drives me forward, drives me to be more and strive for more. Without them supporting me, being excited for me, and giving me a figurative kick in the rear when I am slacking this project would have never been completed. I can't thank my son Xander and my wife Michelle enough, I love you!

Last, I want to thank those who have given me feedback on the many aspects of this project and allowed me to bounce ideas off of you. Tyler Seymour, Daniel Huckaby, and William Grand are just a few of the many who have help me over the years.

Table of Contents

List of Figures

List of Tables

Abstract

OrgTrac: Building an Android Application With Dynamic and Cost Effective Solutions

By

Robert Dale Johnson III

Master of Science in Electrical Engineering

All throughout society mobile devices are now ubiquitous and organizations, businesses, and even individuals with the resources are taking advantage of this by creating a mobile presence allowing them to be within hands reach of their audiences at all times. However, to accomplish this takes quite a bit of time and resources putting those without large resources at a disadvantage to entering the mobile application space. This project will demonstrate a cost-effective approach to building a dynamic Android mobile application that will allow those with limited time and resources to enter into this market and reach their audiences. The project will make use of free and affordable resources allowing the creation of an Android mobile application for any organization, business, or individual. The end product is a step-by-step guide to defining, using, and implementing these resources provided by this project to build and publish an application to the Google Play Store.

1. Introduction

Despite the fact that mobile devices are in the hands of just about everyone and entities of all types are creating for themselves a mobile application presence there is effectively a barrier of resources and expertise keeping under-funded, smaller entities from joining this trend. This effectively makes it impossible for local and smaller groups to be able to build a mobile application presence for the audiences they wish to communicate directly with. Groups from school clubs and teams, to community groups, to non-profit charities, have little to no options for creating a mobile application. The ability of these groups to source the required funds and expertise to accomplish this is highly unlikely.

The objective of OrgTrac is to bridge this gap for these groups, giving them the expertise they need while using cost-effective solutions bringing the ability to build a mobile application within reach of these smaller and less-funded groups. There are many solutions available for gaining expertise in mobile application development but these solutions are usually courses tailored to learning the skills to become a programmer. Many of these courses either address specific development skills or building an entire application from scratch. These courses typically take months to gain the most rudimentary of skills in building an application and do little to show how to build specific types of applications. Additionally, these courses also typically do not address applications being driven by dynamic resources, instead they focus on simple, static application meant solely to show basic concepts. Inevitably making updates to applications like these will require additional coding, building of the application, pushing the new application to the Google Play Store, and users then downloading and installing the new version. Every one of these steps is a process someone has to know and take time to do.

OrgTrac does not intend to remove all these step. In fact, all these steps must be done at least once. What OrgTrac intends to accomplish is minimizing these steps, the knowledge and expertise needed to accomplish them, and limit which need to be repeated in order to create, publish and update a mobile application. OrgTrac will accomplish this by identifying and defining each step of the process then minimizing or completely automating each step. OrgTrac will also attempt to remove the need to repeat steps wherever possible by using other tools or platforms to accomplish the same task. An entity will need the ability to create, publish, and update their mobile application. OrgTrac will accomplish this is several ways, all of which will be defined within a Wiki Walkthrough intended to take any interested entity through all three processes; create, publish, update.

Creating an application is arguably the most difficult of these processes. In order to create the application an entity will need to know how to create a project using Android Studio, program in several programming language including Java/Kotlin and XML, and understand the process of creating the required resources needed for later building the application. Numerous activities related to defining and creating all the elements needed in an Android application also need to be known from using assets such as images and text, to defining and building the User Interface, to defining and implementing navigation between screens all need to be understood in order to create an Android application. OrgTrac will completely remove the need to know the majority of these tasks. It will implement and provide all the source code, assets, navigation logic, user interface, and so on allowing the entities building the application the ability to do so simply by editing a few key values within several files that will identify and distinguish their application from others.

Building the application binary and then publishing it to the store is a slightly less complicated task. Building the binary requires creating a certificate to sign the binary with and instructing Android Studio to build the binary signing it with this certificate. Publishing the application consists primarily of creating an account with the Google Play Store, configuring several values within their system, then uploading the binary generated with Android Studio and the certificate. The project will walk the entity creating the application through this process once again with the goal to provide all the expertise needed minimizing the effort needed.

Inevitably the application will need to be updated to reflect the current state of the entity who built it. This could be as simple as adding or changing text on a single screen, making changes across multiple screens, or even completely removing or adding entire screens. Regardless of the updates nature, this would typically require rebuilding the application after making those changes to the source code, republishing the newly signed application binary to the Google Play Store and then counting on the users of the application to download and install the new version in order to see those changes. This task is likely the most tedious and time consuming task as it requires repeating the building and publishing step every time an update is needed. OrgTrac will provide the ability to bypass the building and publishing of the application each time an update is needed by using a Content Management System (CMS) to define, store, and update the application's content. OrgTrac will define basic components within the CMS that will be used to define the entire application, all its screens, and the navigation between them. The CMS will provide a simple data entry interface allowing the entities both building and updating the application to define its entire content and behavior by using the simple building blocks it provides. Since the CMS and the content defined within it are not within the application source code or binary, updates to the application can be done simply be updating what is in the CMS. Upon the next launch of the it will load those changes directly from the CMS updating itself without the need to rebuild, resign, republish.

Ultimately, the OrgTrac project will be a culmination of all the resources and processes needed to build an application from the very first step to the very last. It will be a combination of resources, tools, coding, and walk-throughs that will give any entity the ability to make a mobile Android application for their audience. The project's goal is to build and provide all these resources and do so in a manner that allows the entities who wish to build their application the ability to focus on process and content without the need to know how to code. Each step of the process will be either completely handled by a piece of OrgTrac or by a walk-thru provided as part of this project. The need for technical skills will be removed and replaced with configuration and data entry. Anything beyond data entry will be handled with targeted, detailed walk-throughs managing the entities process through them in a step-by-step manner once again with the goal to remove the need of technical skills and minimize the knowledge and effort needed. Additionally, at every opportunity free or low cost solutions are used to minimize the cost typically associated with building applications driven by content hosted on servers, "the costs of these apps are [high] and could range anywhere from \$8,000 to \$50,000. [1]"

2. Platform, Tools, and Libraries

The availability of resources and tools used to build and maintain mobile applications is for all intents and purposes infinite. For any given task involved in building an application there are numerous options of tools to accomplish the job. The quality and usability of these tools varies just as much as the quantity of options available. The cost of these tools varies just as much and can range from free to literally millions of dollars to use over time.

At the core of this project's goals is to choose resources and tools that are efficient and cost-effective while insuring that documentation and resources are as available for the tools ensuring they are as usable as possible. Every platform choice, tool consideration, and design principle used throughout OrgTrac was made keeping in mind efficiency, cost, reliability, documentation, platform support, and reusability within the Android ecosystem and others beyond Android for possible future implementation.

2.1 Version-Control System

When developing assets, regardless of whether they are physical or virtual goods, having a safe and reliable location to store your products is paramount. Within software engineering, and more specifically software development, the most important system for storing your product will also store, or track, each version of your product, changes to it, and the users that have access to it. Typically these systems are referred to as version-control systems (VCSs) and are used by nearly every project that includes software development.

For the purposes of OrgTrac there were several VCSs that were considered. These system typically provide a suite of functionality on top of the underlying VCS that includes features such as source code viewing, user access management, issue/defect tracking, documentation support, and velocity graphs just to name a few. When choosing a System and underlying VCS it was important to insure that the system would be widely available, an industry standard within software development, and that it provided support for several of the needs of the OrgTrac project.

When deciding which of these systems to use Git [2] was chosen as the underlying VCS technology best suited to the OrgTrac project with the implementing candidate systems being either Bitbucket [3] or GitHub [4]. Bitbucket and GitHub are among the most highly used systems in the software development industry and both fit the majority of the needs for OrgTrac. Both Bitbucket and GitHub provide a large suite of functionality, support multiple technologies including several different VCSs, and many other advanced features useful to OrgTrac. While both systems implement nearly the same feature set the main differences between the two center around pricing and billing. Ultimately GitHub was the ideal candidate based on the needs and goals of the project. Essentially, GitHub and Bitbucket provide all the same features but GitHub provides free repository hosting for publicly available projects which at this stage in OrgTrac is the better option.

2.2 Android

With any project, or product, picking the correct platform to build it on, and to release it to, can be the deciding factors in the success of the project. When choosing which mobile platform to build OrgTrac on keeping the costs low, expertise required minimal, and ability to find resources high were the deciding factors that needed to be addressed. The Android operating system provides all of these in at least one way.

The Android operating system itself is an opensource initiative meaning that at its core it has been built from the ground up to be used for and by as many people as possible at as little cost as possible to all those involved from the manufacturer of the devices to the customers using them in their day-to-day life. [5] The tools used to create application for the operating system are themselves freely available and compatible with all modern operating systems. The tools for building, publishing, and updating the applications for the Android operating system are all documented thoroughly and made freely available by the company that owns Android as well as many third party sources.

Additionally, Android devices vary greatly in size, cost, and ability meaning nearly anyone has the ability to own a device and consequently use OrgTrac with little investment. Android is also built using well established, and well documented, programming languages and best practices. Tie all these factors in with my extensive experience developing mobile applications for Android in the industry and Android is the best choice for the initial OrgTrac project.

2.3 Android Studio

Android Studio (AS) is the de-facto Integrated Developer Environment (IDE) for developing Android applications for Android. IDEs support the development lifecycle of software engineers providing the tools and resources for the developer to program the application, and typically ultimately build the application. One can think of an IDE to the developer as accounting software would be to an Accountant, it is what they use to accomplish their core responsibilities in their job. AS is where the majority of the effort and logic for OrgTrac will exist and be worked on outside of the Wiki. The code and resources associated with OrgTrac will be created and edited using AS and the binary for publishing to the store and installing on the Android devices will be generated using AS.

While those using OrgTrac and the Wiki to build the application will not need to use the majority of the features of AS it will be used in several steps of the process since it provides the most intuitive interface for modifying and building the application. AS will provide the easiest place to update values specific to those building the project in the future and has setup wizard that walk you through building the Android application binary. AS will be heavily used in the process of creating the OrgTrac application and source code and will effectively be the most used tool other than Contentful [6]. The Wiki that is part of this project deliverable will also make use of the setup wizards provided within AS for common tasks such as generating signature certificates and building the application binary.

2.4 Google Play Store

There are two primary ways to distribute Android application binaries for installation on devices; through digital storefronts such as Google Play or by a technique called sideloading. Both of these methods for distributing applications have their respective positives and negatives so weighing the options and making the correct choice must take more than simply being cost-effective and easy.

Sideloading is the simplest form for distribution because you simply need to build the application binary and make it available for download somewhere. However, this method is insecure and requires users installing the application to change security settings on their phone. While sideloading is easy and requires less process on the development side it doesn't provide a robust and reliable distribution path. Sideloading requires users to disable security features on their devices that will leave the device unnecessarily open to many other potential security risks. While OrgTracs goal is to make things as cost-effective and efficient as possible the project chooses to do so without sacrificing process and best practices.

The process of distributing Android application binaries through digital storefronts is a far more involved process for the person publishing the application. In order to publish applications on the Google Play Store you must register for an Android developer account and pay $25 registration fee. Additionally there are many different steps and several required pieces of information that need to be defined in the system in order to publish the application binary. Specific setup process, assets, and other resources have to be made available in the system as well. Establishing an android developer account and providing all the required information to the system prior to being able to publish the application can be a tedious endeavor, but it only has to be done once. By taking these steps it allows your application to be published to the biggest Android application digital storefront and is done through the most robust, reliable, and secure digital storefront available for Android. Despite this not being the most cost-effective option since there are free options available it is only an initial $25 registration fee but provides so much more that is both better for the publisher and the user. Publishers know they are providing a secure, reliable, and safe method of publishing their application to their users and the users can install the application without having to make any sacrifices to the security of their device and do so from the Google Play Store where they likely get all their applications. The Google Play Store also provides some helpful insight into application metrics including installs, user retention, feedback, and error reporting. Any complications involved with publishing through the digital storefront will be simplified and explained within OrgTrac's wiki effectively removing most of the difficulties of publishing this way.

2.5 Content Management System (CMS)

The true power of the OrgTrac project will be its ability to be defined and updated dynamically from a web-hosted source. Having the ability to host the content of the application on the internet and then the application access it in order to build what the end audience sees on their mobile device is what allows OrgTrac to accomplish many of goals of this project. There are many different ways in which this data could be defined and hosted on the internet. There are several options available in order to achieve the ability to host the applications content and subsequently load that content into the application.

One such solution could be to simply define all the content for the application in a file, think of this as a Microsoft Word document file that the application can read, and host it anywhere on the internet that can be reached by a browser or in this case the application code. In order to do this you'd need to know how to define the file and its content, know what to use to create and update the file, and where and how to upload it to the internet. These steps may seem simple but if you don't define the file just right, if you make any minor mistake in text or formatting, the application won't know how to use the file. In order to create and update the file you will need to know what software to use and how to use it to edit the content. In order to upload the file you'll need to know some type of file hosting and uploading mechanism that will make that file available on the internet.

This by nature is too unreliable and requires the user to know quite a bit about web-hosting and the content format making it an unwieldy solution for OrgTrac

Another solution would be to create an application program interface (API) that provides the ability to create, update, and upload the content needed. The API would itself need to be created and hosted on the internet. The advantage of the API is once complete it would handle all the aspects of defining, updating, and uploading content to the internet providing the ability for the user to update the content using the API while the API would insure all the formatting, updating, etc of what the application will use is properly created and made available. This solution would make the adding and updating of content mostly intuitive for the user but the API itself would first need to be created and hosted on the internet meaning even more knowledge of web-hosting and internet technologies again making for a solution too unwieldy for OrgTrac.

The correct solution for OrgTrac will ease the process of defining the data, ease the creation and updating of the data and require minimal knowledge of web-hosting and other internet technologies. It will accomplish these using an intuitive and easily managed system that will provide the formatting and definition of the data and make creating and updating it as easy as filling out a standard information form. Using a content management system (CMS) is how OrgTrac will accomplish this. When choosing a CMS many things most be considered as there are numerous options available. A CMS can come in many forms and many are built for very specific purposes such as defining content to be used in an e-commerce store. Some CMSs are meant to be used to build entire websites while others may be simply a way to store customer information for a call center. The majority of CMSs are meant for a specific purpose, within a specific domain, with specific content options available.

Since OrgTrac and its data models do not follow any standard industry formatting a CMS tailored to OrgTracs needs does not exist so a type of CMS called a headless CMS will need to be used. Instead of defining specific models and data tailored to one use a headless CMS is detached from any specific use case and consists only of the raw ability to define and create data meant to be able to be used by any system [7]. This gives the ability to define data models in any way the user wishes that can then be created and updated easily through forms .Later this data can consumed by any platform, tool, or resource that wishes to use it. This gives OrgTrac to ability to easily define its data models through an intuitive system and those models are then translated into easy to use forms that the system will use to curate data it will make available to the application.

This project will use Contentful [6] as its headless CMS choice. Contentful provides all the features needed from a headless CMS and has a very robust suite of tools for using Contentful within other systems including Android.

2.6 Libraries

When developing software, especially when time and resources are limited, it’s important to use tools and libraries that are both robust and reliable. With the open nature of Android and its usage of the Java programming language there is the ability to choose from a long history of library and tool support that has existed and been maturing for decades. Combining these long available resources, and the expertise of those who created them, with the growing and maturing resources for Android and it becomes a choose of preference and availability among many available libraries.

When deciding what tools and libraries to use it is at times simply a natural extension of other tools you are using. There is a large suite of libraries and resources made available by the developers who maintain Android itself that come in the form of the Android Support Libraries that build upon and extend the core features available to all Android applications which OrgTrac and this project will make great use of [8]. These tools and libraries include Android Studio itself as well as many of the resources it takes advantage of. One such tool is Gradle a dependency and build tool that Android Studio uses to resolve and retrieve other libraries and their dependencies. Contenful and the choice to use it for this project was heavily weighted on the libraries that Contentful produces for many major platforms including Android [9]. Contentful also provides a robust syncing and storing library called Vault that gives the added benefits of persistent storage and model declarations that allows for a more robust and reliable integration with Contentful that also limits network traffic when using the application and retrieving updates [10].

Several other libraries will also be used within this project. These libraries will provide robust and reliable support while developing the applications source code, inevitably make the process of developing the application more efficient and reliable. OrgTrac will make use of several Kotlin Language extensions libraries, including KAndroid and Anko, built to be used with Android development tasks [11][12]. These libraries are used to avoid unnecessary boiler plate code and help avoid common mistakes that are easily made but that the library functions themselves implement properly. The project will also make use of two libraries, RxJava and Dagger, which help to implement reactive and reusable architecture within the application code [13][14][15].

The library logging library Timber will also be used as it eases development and debugging takes since it provides robust and easy to use logging mechanisms [16]. A common issue, or pain point, when developing Android applications is the use of images hosted on the internet within the all. There are many ways to accomplish this as well as many ways to do it inefficiently and incorrectly so this project will use the library Picasso [17].

3. Project Approach

The catalyst for this project is a desire to provide the ability to create an Android application to those people and organizations who do not have the time, resources, expertise, or financial means necessary to do so. There are innumerable choices available when building an application presence and this projects goal is to make available an all-in-one source for going through the process using tools and resources readily available to anyone interested in the endeavor. Recognizing the nature of the resources available, the potential cost to use these resources, and the need for expertise in order to build an application is the core focus of this project. OrgTrac will make use of the tools this project sees as most fit for the task.

3.1 Project Goal

Since the ability to create an Android application obviously exists and can be done by those with the time, resources, and expertise needed the goal of this project isn't to simply create an Android application. Instead, the goal is to provide those without the time, resources, and expertise normally required to build an Android application the knowledge, resources, and instructions to do so. The project is intended as a proof of concept application and accompanied wiki walk-thru, or tutorial, that accomplishes this task. The core expectation of this project is to lower the barrier of entry to building an application in such a way that someone with little or no technical skill can follow and use this projects deliverable to publish an Android application.

In the effort to achieve this goal the OrgTrac project makes use of tools and resources that are readily available across operating system platforms. The tools used are by their nature and expected use easy to work with and the project wiki walks the user through difficult or complicated steps. Areas that require a deep expertise such as coding software, interacting with server APIs, and defining model specifications are completely handled by the project and the tools it makes use of effectively eliminating the difficulties associated with those task.

3.2 GitHub

At the core of this project is the application source code and the wiki that will be used to walk the user through the process of creating and publishing the application for their organization. A reliable, easy to update, and easy to access solution for storing and hosting the source code and wiki is paramount to the success of this project.

For these reasons GitHub is the backbone of the project (https://github.com/rojoiii/a-app-orgtrac). All the source code and documentation, including the wiki, will be managed and/or hosted in GitHub. The feature at the core of GitHub is using it to store Git repositories of your source code which OrgTrac is doing. All source code, assets, etc related to the writing and building of the application are stored in GitHub. This allows for efficient writing and updating of code within the repository with all the advantages of Git, such as history and versioning, allowing for the code to safely be stored and possibly updated in the future. GitHub is also globally available and will provide the mechanisms for users of this project and OrgTrac to access and download the code to their machines during the process of building their own application.

Additionally, GitHub is free to use for open source projects and provides tools for managing the OrgTrac source code repository easily through a Graphical User Interface (GUI) rather than through the command line which is Git’s default interface. This tool is the GitHub Desktop application [18]. This project uses the GitHub Desktop application to fork the application code and download it to the users machine so they have the ability to modify it for their organization. The process of obtaining and using the GitHub Desktop application to retrieve the application source code is defined within the wiki.

Using GitHub allows the project and those wishing to implement it for their own organization the ability to do so at no cost. The GitHub Desktop application is also free to use and provides an intuitive, easy to use, tool for managing the OrgTrac projects source code and other resources. This also extends these ability and no cost to those using OrgTrac to build their own Android application for their organization.

3.3 Wiki

Other than GitHub the most important aspect of this project is the wiki (https://github.com/rojoiii/a-app-orgtrac/wiki). The wiki defines in detail all the tasks necessary to take the project from start to finish walking the user through each necessary step. When it makes sense to the wiki provides links to external resources and walk throughs defined elsewhere that relate to the tasks need. Since the tools and technologies used in the project were chosen because of their already existing resources and efficiency the wiki routinely relies upon these resource either for brevity within the wiki itself or because the already available resources are sufficient.

The wiki consists of three major sections that corresponding to the highest level tasks needed in taking OrgTrac from start to finish. These major sections are: Get Your tools, Register for Online Tools, and Using the Tools. Each of these steps within the wiki have multiple parts and when things are unclear or ambiguous images, charts, and videos are provided to clarify the process. Each task is broken down into easy to follow tasks that intend to not overwhelm the user with too little or too much detail. Several tasks will also be handled using scripts, such as importing Contentful models and data, that will enable the user to more easily complete tasks that might otherwise require more knowledge or expertise in other technologies. The wiki is a one stop shop for all the resources and tasks necessary to bring the application from concept to published in the store.

3.4 Android Studio

Android Studio (AS), and subsequently Contentful, is where most of the users time will be spent outside of the wiki. Once the source code has been downloaded from GitHub to the users machine using GitHub Desktop the project will be opened using AS. There will be several major tasks accomplished in AS which break down into three major efforts: updating configuration for the users organization, running the application in debug mode on an emulator, building the Android application binary for publishing. These two tasks will take the OrgTrac codebase from its configuration as a sample project, which will be the default setup when downloaded from GitHub.

The first task AS will be used for is updating values within the OrgTrac AS project. In order to customize and eventually publish OrgTrac as an application for the user's organization several steps must be taken to adjust OrgTrac to the new organization. All of these adjustments will be explained within the wiki and include at least the following: update the package name, update the application name, update the Contentful API token and Space ID. There will also be several other options that can optionally be updated to customize OrgTrac for the user's organization. These optional updates will also be explained in the wiki and include at least the following: update the application colors, update data retrieval error messaging, update the application icon.

The second task AS will be used for is running the application on an emulated device. AS can be used in two ways to create and use the application binary. The first of these is running the application in debug mode allowing the user to run the organization's application on a virtual device on their computer prior to generating the binary for publication on the Google Play store. Once the project is downloaded and opened in AS the user will be able to run the demo application on the emulated device. While working through the wiki and making updates to the OrgTrac application for their organization users will be able to run the application as many times as they want allowing them to see the changes as they make them. This will give the user the ability to see their changes, making all the necessary changes for their organization and verify that they work and show as expected in their application prior to generating the binary and publishing it to the Google Play Store. This will allow them to view the application as someone in their audience would see it, giving them the ability to fully customize the application and see how their changes look before anyone else has a chance to see the content.

The third task that AS will be used for is generating the application binary. The user will do so once they have made all the appropriate changes within AS and Contentful and feel confident that the application is running and looking as they expect. The user will first generate a signature certificate that will be used to digitally sign the application binary when it is generated. AS has a simple wizard for generating the certificate and another wizard for using the certificate to generate and sign the application binary. The wiki will walk the user through each of these processes. The certificate and the credentials within it act like a lock and key to the application binary and must be saved and stored securely since they are used to validate that the application is valid within the Google Play Store. When updates need to be made to the application binary this certificate will need to be used to generate the new application binary otherwise it will be rejected in the Google Play Store since any future publication of binaries to the store must have the same signature as the original update. The user is advised to store the certificate and credentials in a secure way, preferably not within GitHub unless they use a private repository.

3.5 Contentful

Contentful is where all of the data that drives the applications content and layout will reside. Within Contentful will be the definitions of the content models that are made available for display within the application. Contentful uses these content models to present the user with easy to create and update forms that they will use to create all the content for the organization's application. What the content models are, what restrictions they have, and what types of fields they have available are defined by this project and will be discussed later in the Architecture section. This project has implemented these content models within Contentful. The wiki will walk the user through the process of importing these models into their organization’s Contentful space as well as import example data if they wish to do so. This will be done using a sample Space defined in Contentful as part of this project and the wiki.

By using Contentful to define the structure and content within the OrgTrac applications users will have nearly infinite options to define the way in which their application’s data is presented. There are three major content models that correlate to pages or screens in the application and nine content models that correlate to content that can be displayed on a page. There is also a Link content model that provides the linking between a page and the content that can be displayed within it. These three page models and nine content models correlate directly with the models defined in the Architecture of the OrgTrac application and hence define all the possible options available to the user for building the application structure and content.

Within Contentful the user will define the content for the loading page, landing page, and main pages following the simple forms provided by Contentful. There is only one loading and one landing page defined per application. The loading page is most accurately described as a loading splash screen that will show while the rest of the content for the application is synchronized from the Contentful web server to the mobile device. The landing page is the main menu of the application and links to main pages that are effectively all the screens of the application. Each main page is a combination of the other nine content models in any order or combination the user wishes to define. Using the nine content models the user can define a main page that is made up of any combination of content giving them the ability to create content within their application in nearly any way they wish. Any content model that will be displayed in a main page must have a corresponding Link Reference that will be used to link content to the main page.

The process of creating the three main pages, the nine content models, and the ability to link them is shown within the wiki and can be mirrored using the sample data provided with the project.

3.6 Architecture

The architecture of this project and OrgTrac specifically is built around the definitions of the three page models and nine content models. The structure of the code and logic defining how it is used is also built around these models. The project is built using the Kotlin programming language since it is the new standard for Android

Development and makes use of many Java and other Android specific languages and tools.

Object dependencies within the code are created and used using a Dependency Injection framework called Daggar 2 [15]. This allows for the creation of all object dependencies in one source code file, AppModule.kt. All dependencies are defined here whether or not they are specific to OrgTrac's implementation or simply defining instances of included library classes. This allows for all dependencies and their relations to be defined in a single location that is more easily managed and maintained especially as dependencies change. Additionally, when using these dependencies a simple annotation declaration is used to instantiate instances within implementing classes making the use of these dependencies across many classes within the applications source code simple and error free because it is done in the single source code file.

Data loading and resource intensive tasks are done asynchronously so that the application does not appear to stutter or skip since the tasks of displaying the UI and completing these time and resource intensive tasks are accomplished on separate thread. In order to accomplish this asynchronous nature OrgTrac makes use of RxJava [13] and RxAndroid [14] which provides the facilities to define what tasks are done, on which threads, in what order, and how to respond should the task execute successfully or with errors. This allows the application to display states on the pages while the heavy tasks execute as well as respond when those tasks complete in a way that feels both natural and fast to the user within some of the lag and jitter that can sometimes be associated with tasks within applications that are complicated, this is especially true when transitioning between screens.

All of the data that is synchronized from Contentful is stored locally on the device in an SQLite database using Vault [10]. Vault and Contentful work together to download the content the user defines within Contentful to a local database file. All the applications content is stored within the database file using vault and upon each subsequent launch of the application is updated with any changes since the last run of the application. This means that the application downloads the entire set of data only once and from then on only downloads the changes allowing the application to use the least amount of data possible. This also means that all content other than images or content linked to outside of the application are available immediately. Instead of needing to constantly ask web servers what the content is and downloading it before being able to see it the content for each page is always ready for display from the database meaning the application experience and load times are incredibly fast.

Images are also downloaded and stored on the devices whenever possible, caching them for future use following standard practices and memory saving techniques mostly through the usage of Picasso [17]. Picasso provides the application with the ability to store images within the application in a similar fashion similar to the way Vault stores the rest of the content. Picasso will load and cache images on their first display within the applications pages with all subsequent usages being loaded from memory rather than from the internet. This too provides abilities similar to Vault allowing images to be loaded using minimal internet data with subsequent displays loading faster from memory.

The rest of the architecture is built around the three page models and nine content models. The three page models are: Loading, Landing, Mian. The nine content models consist of five templates and four elements: Article, Contact Info, Location, Profile, Web and Image, Link, Text, YouTube Video respectively. The application stores the Contentful data using Vault and the models defined as part of implementing the SQLite database with Vault. Once all this data is stored in the SQLite database it is translated into memory using the OrgTrac models defined as part of this projects specification. The Vault database models and OrgTrac models are very similar, however, the Vault models include additional fields that are not part of the OrgTrac specification and purely exist since they are needed for Contentful and Vault. The OrgTrac models are used throughout the application while the Vault database models are not. The OrgTrac models are defined to include only exactly what is needed in order for the OrgTrac application to properly define structure and content. These models are defined separately from each other and are translated using the DataManager.kt class file. Ideally, OrgTrac will eventually support different CMS sources so the architecture has separated the loading, storing, and updating of Contentful/Vault data from the data and models used in the UI. This will allow future efforts to implement additional CMSs more simply without having to adjust the code specific to the UI and business logic since it is built using solely the OrgTrac data models which are independent of Contentful and Vault.

3.7 Approach

The overall approach of this project includes taking a user from little or no knowledge of building an application to having an application published in the Google Play Store and doing so while incurring as little cost as possible. At the core of accomplishing this is a Wiki hosted in GitHub (https://github.com/rojoiii/a-app-orgtrac/wiki) that walks the user through the entire process. The wiki presents the users with all the information and tools they need to take the OrgTrac source code and sample data provided through this project and using it to create their own mobile application. The Wiki accomplishes this through three major sections: Getting Your Tools, Registering for Online Tools, Using Your Tools.

Within the first section, Getting Your Tools, the user is walked through the process of obtaining the tools they will need on their computer. These tools center around the applications needed to download the OrgTrac application source code and modifying it to suit their organization, ultimately allowing the user to generate the binary needed for publishing to the Google Play Store. The two tools the user will need are Android Studio and GitHub Desktop. The user is shown how to get these applications, install them, and configure them such that they can be used to take OrgTrac from being a sample application to an application for their organization.

Within the second section, Registering For Online Tools, the user is walked through the process of creating and setting up accounts with the various online tools they will make use of. These tools do everything from giving them access to the source code they will need to build the project to the ability to publish the application on the Google Play Store. The three accounts that will need to be setup include GitHub, Google Play Store with a Developer Account, and Contentful. The user will be shown what information is needed and what processes need to be complete in order to use these tools to build OrgTrac for their organization. How these are used and what the user will use them for in order to build their organizations application is discussed in the third section of the wiki.

Within the third section, Using the Tools, the user is walked through each step required to build the application for their organization using the tools and accounts setup earlier. This section of the wiki is where the majority of the user’s time will be spent as it will walk them through the process step by step. Initially the user is shown how to retrieve the source code from GitHub using the GitHub Desktop application. This is followed by showing the user how to import the models and data from the sample Contentful space into their organizations space. The basics of what data is required in Contentful and how to update and modify the sample data to suite their application is also provided. Once Contentful is in a state the user is happy with they are shown how to open the source code using Android Studio. They will be shown how launch the sample application provided in the OrgTrac source code using the Android emulator. From here the user will use AS to update the sample OrgTrac application source code within Android Studio to match their own organization. Once the migration from OrgTrac sample data application to their organizations data is complete they will be able to launch their application in the Android emulator. The user will then be walked through how and why to generate a signature certificate with advice on how to store it securely. Once the certificate is generated it will be used within AS to generate a signed Android binary file for publishing in the Google Play Store. Finally the user is shown the process of uploading the binary to the Google Play Store and their organizations application will officially be ready for their audience.

4. Project Creation

This project has created several large efforts that are intimately related but also completely separate efforts and deliverables from each other. The core of how this project is brought together is through the wiki that ties these desperate efforts together in a resource that provides the user with everything they need to build OrgTrac into their own application. This project includes the creation of application source code, generating the data driving that application, and a resource for defining how the user puts these pieces together. These major components, the source code, the data, and the wiki are at the core of creating this project and accomplishing its goals.

Since the wiki provides a definitive resource on how to use these three pieces together in order to build OrgTrac the focus when creating them hinged on making them as easy to use and as intuitive to follow as possible. Many decisions were made along the way to minimize the complexity of interaction with these pieces in an effort to ease the process for the user. When creating each piece of the project I was able to make use of knowledge I already had with developing Android application, using Contentful, and defining Wiki content which allowed me to focus on the quality of the efforts and bringing them together rather than having to completely learn new tools and technologies.

4.1 The Source Code

The application source code was written using Android Studio and the Android SDK while making use of several well-known and well-defined libraries. The majority of the code is written in Kotlin with Android specific content definiitons being in XML. The source code is broken down into several major components: data classes, activities, layout resources, assets, and helper classes. All of these components follow Android paradigm specific practices and as well as patterns that I have found intuitive and useful through experience.

The data classes are arguably the most important aspect of the source code since they are used to define and interact with the data for which OrgTrac is built around. There are two specific data model packages within OrgTrac one that is specific to Contentful and another specific to OrgTrac itself. The Contentful specific models are defined in the file _contentful.kt. They define the models used by Vault to both download and parse the data from the Contentful servers and to store the data within the applications SQLite database. These models follow the specification defined by Contentful and Vault while implementing the data required by the OrgTrac application models defined in the file _orgtrac.kt . The data models specific to OrgTrac define the necessary data needed for use throughout the application for building the structure and content of the application. The data models defined for Vault and Contentful are translated into the OrgTrac models for use throughout the rest of the application using the DataManager.kt helper class. The OrgTrac and Contenful models consist of the following: Loading Page, Landing Page, Mian Page, Article Template, Contact Info Template, Location Template, Profile Template, Web Template, Image Element, Link Element, Text Element, YouTube Video Element. These models, their fields, and their relationships define the structure of the entire application and can be understood by reviewing the class diagrams in Figure 1 and Figure 2Figure 2 - Contentful/Vault Model Class Diagram.

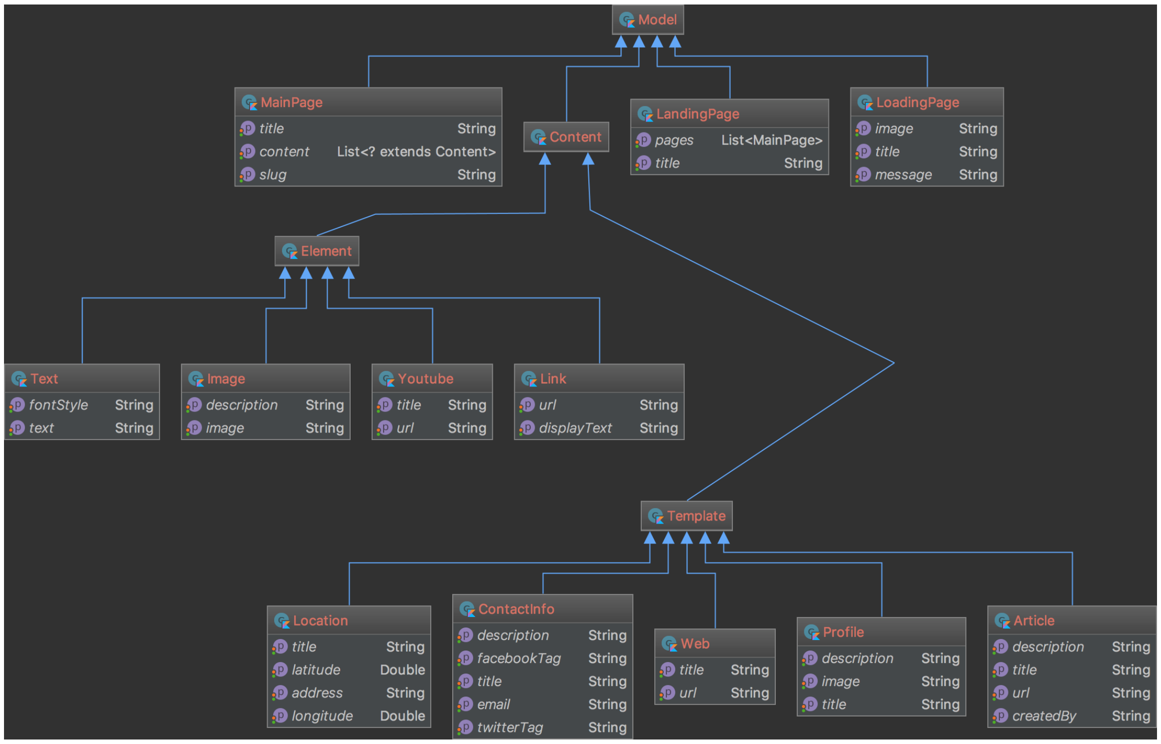


Figure 1 - OrgTrac Model Class Diagram

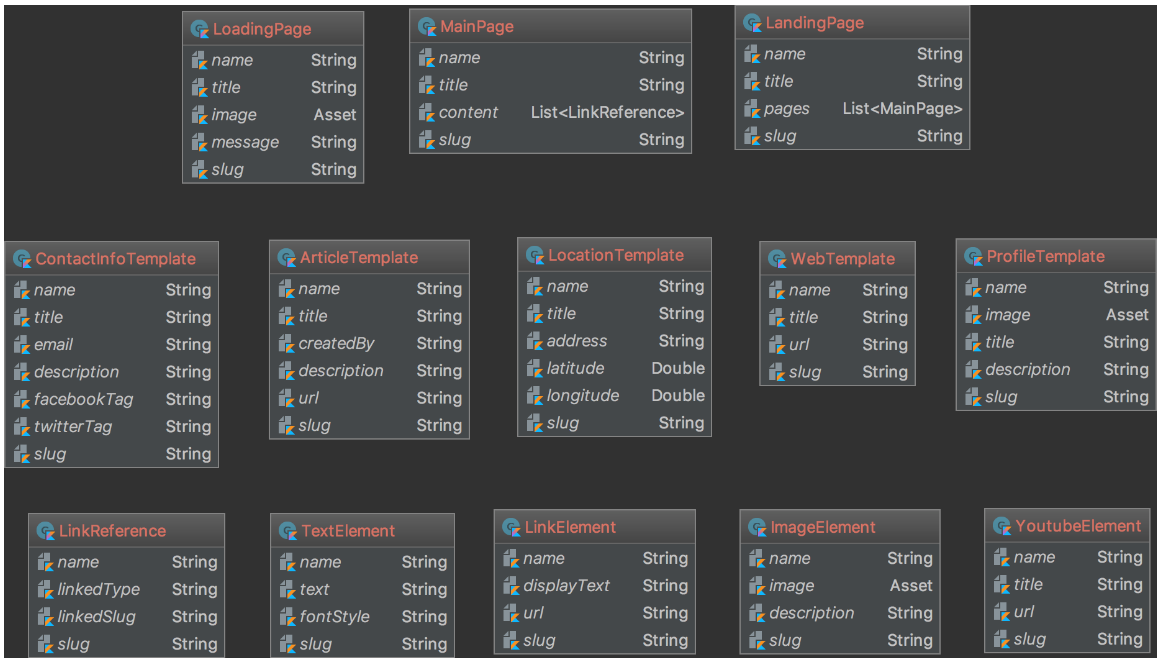
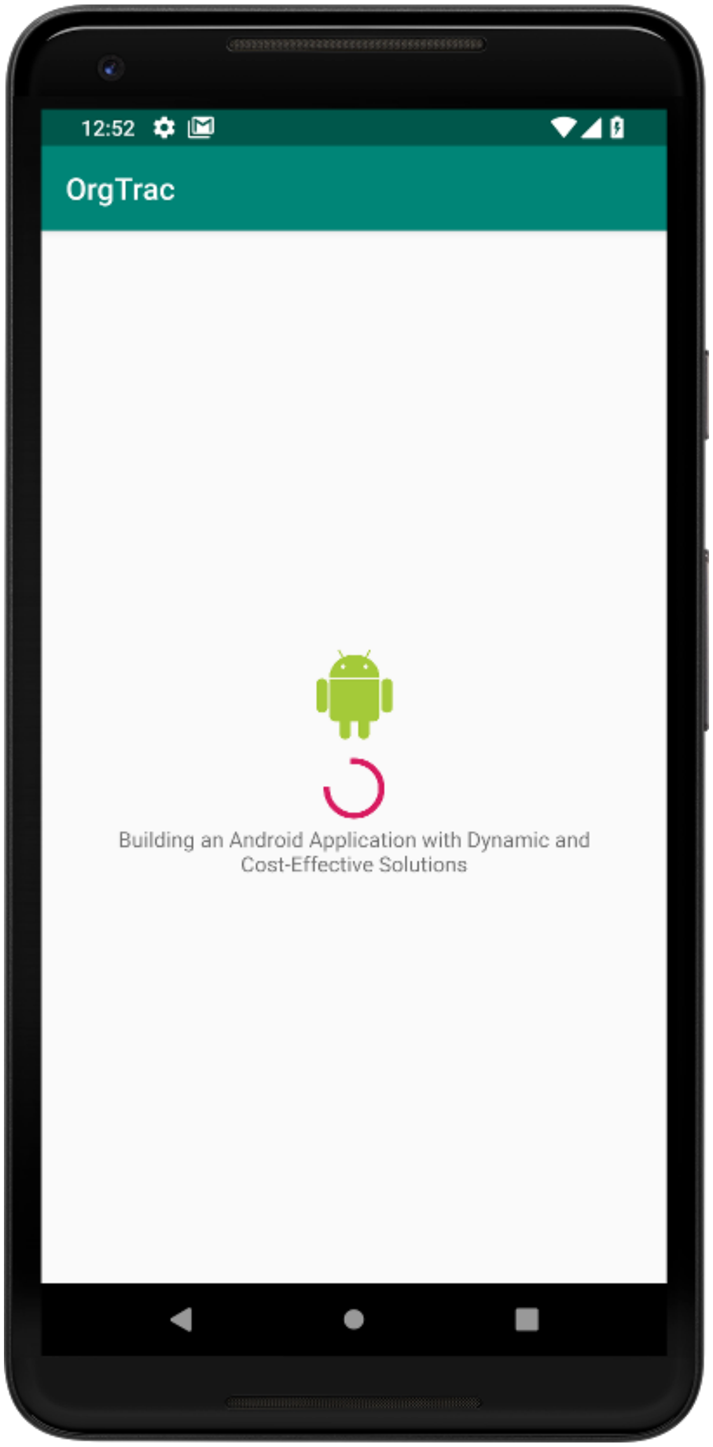


Figure 2 - Contentful/Vault Model Class Diagram

The activities classes in the project correspond to screens within Android applications and are built using the data in the corresponding page models defined for OrgTrac and in

Contentful. The entire application consists of building the three activities that correspond to the Loading, Landing, and Main page models. Within the project these activities are defined within the following classes: LoadingActivity.kt, LandingActivity.kt, and MainActivity.kt. Each of these activates uses the DataManager dependency through Dagger to access the models specific to them and uses those models to build the screen the application will show. Each of these activities also has an associated layout file that defines the structure and layout of the screen, defining how and where the models' data is displayed on screen. The layout files for the loading, landing, and main activities are activity_loading.xml, activity_landing.xml, and activity_main.xml respectively. Examples of how the layout of the loading, landing, and main activities as they will be shown in the application are shown in Figure 3, Figure 4, and Figure 5 respectively.



Building an Android Application with Dynamic and Cost-Effective Solutions

Figure 3 - Loading Page Screenshot

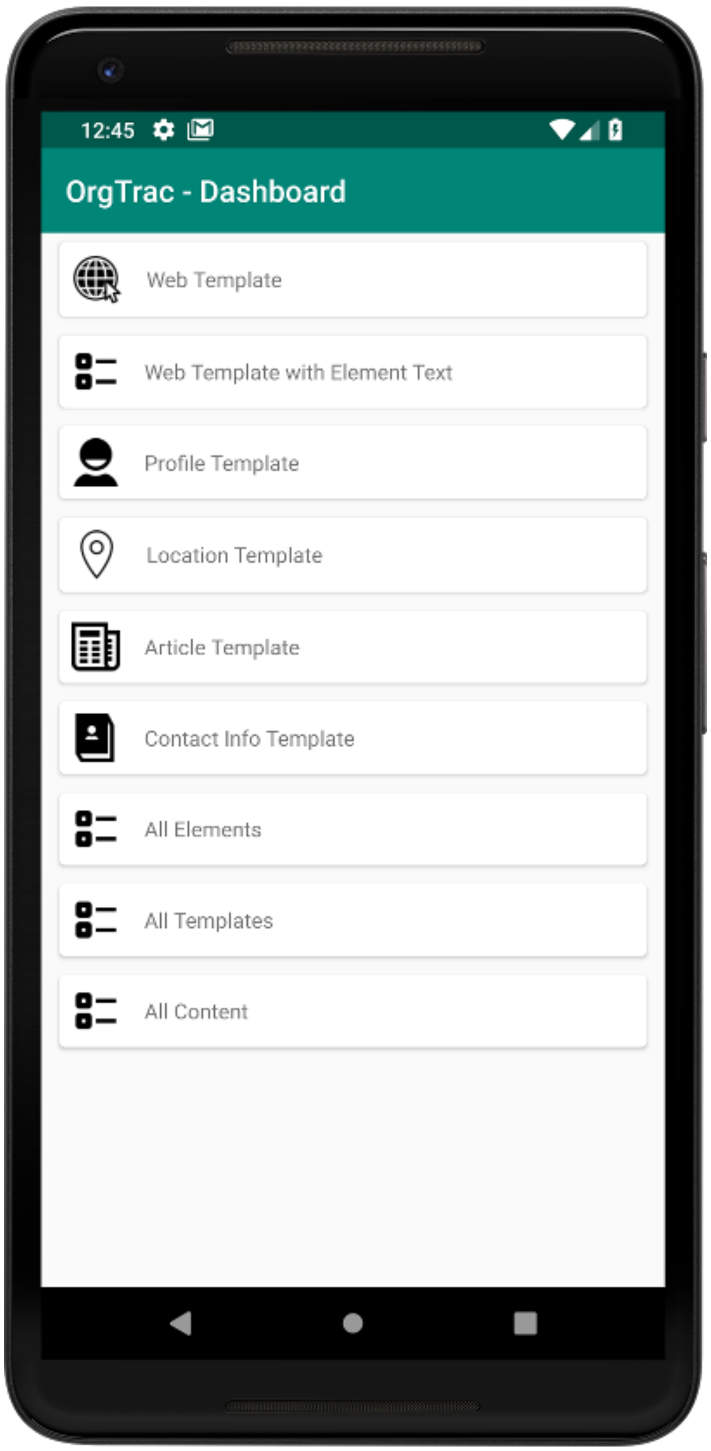
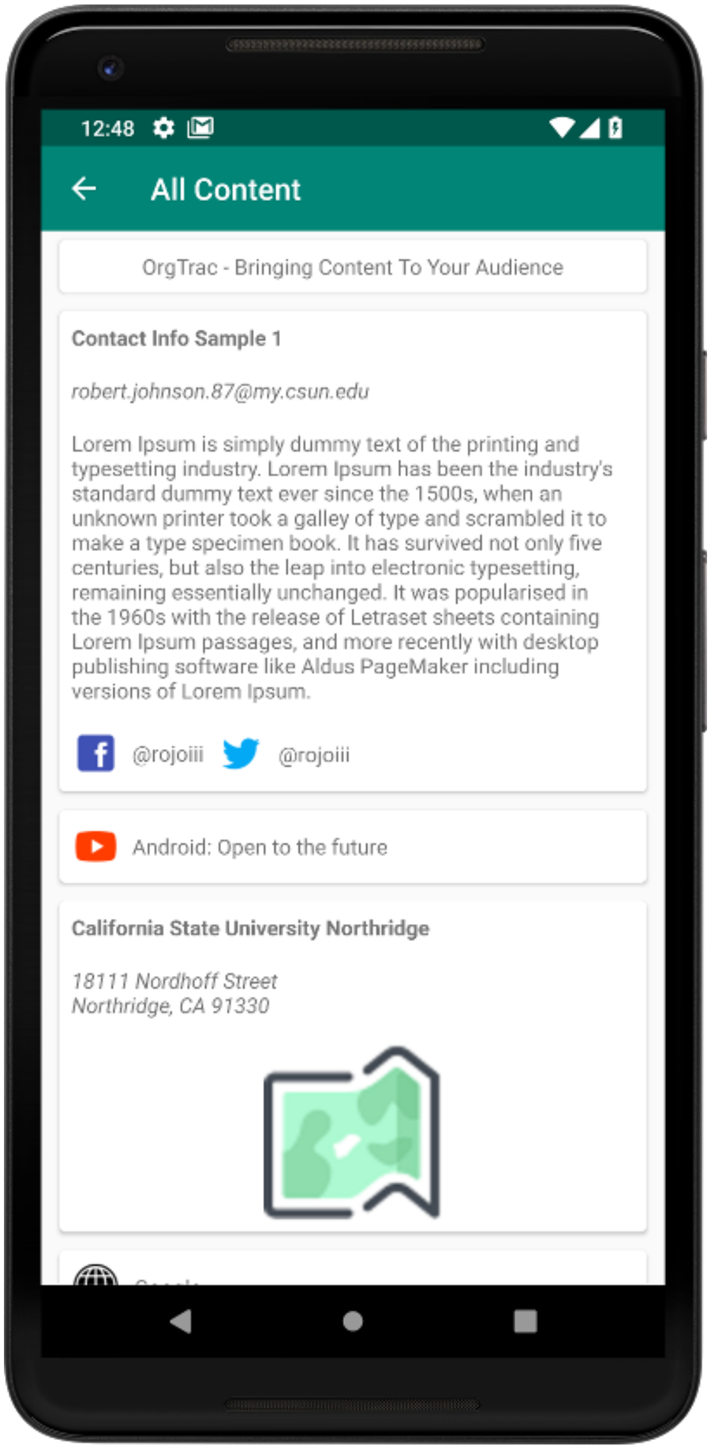


Figure 4 - Landing Page Screenshot


12:48

All Content

OrgTrac - Bringing Content To Your Audience

Contact Info Sample 1

robert.johnson.87@my.csun.edu

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

@rojoiii @rojoiii

Android: Open to the future

California State University Northridge

18111 Nordhoff Street
Northridge, CA 91330

Figure 5 - Main Page Screenshot

The loading activity presents the attributes defined in the LoadingPage model. This consists of displaying the title, image, and a message attribute to the application user. The sole purpose of the loading activity is to load and prepare all the data required for the application. When this screen is shown all the data is synchronized from Contentful using Vault to the applications data storage and then immediately translated into the OrgTrac models that the application uses in the activities to build its UI. The loading activity uses the DataManager to accomplish the loading and translating of the application data. In turn the DataManager makes use of the Contentful and Vault libraries to download and store the data before translating it using methods in the associated OrgTrac models that themselves contain the logic necessary to translate the Contentful data models to their own. Once the Contentful data is stored in the database and translated into OrgTrac data stored in memory the loading activity transitions to the landing activity.

The landing activity presents the attributes defined in the LandingPage model. This consists of displaying the title and a list of main pages that the landing page can transition to. The sole purpose of the landing page is to present to the application user all the content available in the application. When the screen is shown the activity uses the list of MainPage models defined in the LoadingPage model to display a list of buttons that when clicked will transition from the landing activity to the associated main activity. How these buttons are displayed on screen is defined in the layout file activity_landing_item.xml. The landing activity uses the list MainPage models to create a button for each that is then displayed to the user. The landing activity also includes logic for showing icons within each button that are intended to elaborate on what content is within each MainPage.

Different icons are shown depending on the content within the MainPage. A breakdown of these icons and the content they signify are show in Table 1. Once the user selection a button corresponding to a MainPage the landing activity transtions to the main activity associated with that button.

Table 1 - Landing Page Icons

🌐	Web Template Content
👤	Profile Template Content
📍	Location Template Content
📰	Article Template Content
📇	Contact Info Template Content
☰	Mixed Content

The main activity presents the attributes defined in the MainPage model. This consists of displaying the title and a list of content defined for the MainPage being displayed. The sole purpose of the main page is to present to the application user all the content defined for the page being displayed. As with the landing activity the main activity uses layout files to define how the different content pieces are displayed on screen. The main activity will use the templates and elements defined in the content list with their associated layout files to build all content displayed on screen. The main activity also includes action logic for some of the content types shown including actions for the web, location, article and contact info templates and the YouTube and Link elements.

The remaining files include configuration files that either define configuration values for the Android application itself, the Android Operating System, or for tools and libraries the application uses. The most important of these files are the ones that the user will need to modify to customize the OrgTrac application to their own organization. These files include colors.xml, string.xml, contentful.xml, and build.gradle. The colors.xml file contains the three main colors that define the look of the application when ran on a device, the user is able to change these to any color values they like and the entire application will be updated to use those colors. The string.xml file contains several string values that are static and do not change for the application. The user will be able to change these to values they wish to see in the application. The most important string to change is the string named “app_name” since it defines the name that the user will see under the icon once the application is installed. The color values and all the string values except “app_name” are optional to change. The contentful.xml file contains the two values that the application uses to connect to Contentful, these most be updated so the application download all the data needed to build the organization’s application. The build.gradle file contains many very of the most important configuration values for the application, however, only “applicationId” must be changed. This is the unique package name that the Google Play Store will use to distinguish the organization’s application from all the other applications in the store. This value must be changed and must be unique. Typically it’s a reverse url, for example if OrgTrac had a website url hosted on CSUN it might be orgtrac.csun.edu so the “applicationId” would be edu.csun.orgtrac as it is for the OrgTrac sample application. This value and what it should be is completely arbitrary, simply needing to be unique, but by convention follows this example.

4.2 The Data

The data available to the application was designed and implemented to be as simple as possible while facilitating ease of use. All the content was defined with the intention to give the user the ability to easily define data while giving them the power to use that data in any way they wish. The basic building elements are Text, Image, YouTube, and Link. These are purposely simple, making them easy to understand, define, and use in the application. The templates a meant to provide a simple combination of these basic elements that correspond to typical use cases associated with application allowing the user to handle common use case with a single piece of data. The templates build into OrgTrac are Web, Profile, Location, Article, and ContactInfo. Each of these templates correspond to entire screens, or subsections of screens, that users are typically used to seeing in nearly all applications. Both the basic elemetns and the templates are intended to make it as easy and as simple as possible to combine them into whatever application experience the organization may need. Each can stand alone on a main activity or in combination with any other, including as many of each type as needed. The data was also kept as simple as possible allowing the models to be duplicated in systems like Contentful that in turn can easily manage the creation, storage, and downloading of the models.

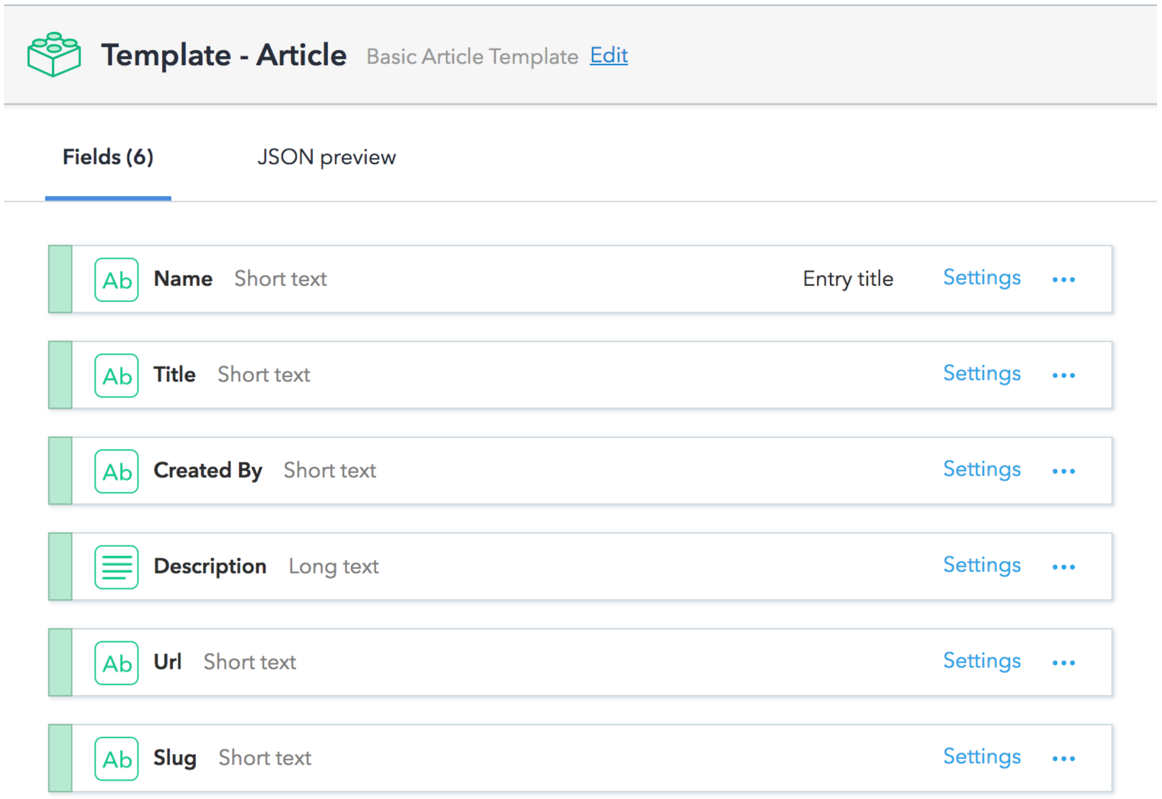
Contentful was chosen as the CMS for this project since the creation of the models and the data within Contentful is rather simple consisting primarily of defining fields for the models and what the restrictions are for those fields. Each model defined within

OrgTrac has a corresponding model defined within Contentful. The OrgTrac models and their corresponding models in Contentful are show in Table 2.

Table 2 - Contentful vs. OrgTrac Models

Contentful.Model.LoadingPage	OrgTrac.Model.LoadingPage
Contentful.Model.LandingPage	OrgTrac.Model.LandingPage
Contentful.Model.MainPage	OrgTrac.Model.MainPage
Contentful.Model.WebTemplate	OrgTrac.Model.Content.Template.Wed
Contentful.Model.ProfileTemplate	OrgTrac.Model.Content.Template.Profile
Contentful.Model.LocationTemplate	OrgTrac.Model.Content.Template.Location
Contentful.Model.ArticleTemplate	OrgTrac.Model.Content.Template.Article
Contentful.Model.ConteactInfoTemplate	OrgTrac.Model.Content.Template.ContactInfo
Contentful.Model.TextElement	OrgTrac.Model.Content.Element.Text
Contentful.Model.ImageElement	OrgTrac.Model.Content.Element.Image
Contentful.Model.YoutubeElement	OrgTrac.Model.Content.Element.Youtube
Contentful.Model.LinkElement	OrgTrac.Model.Content.Element.Link

Contentful then uses these models to provide the user with simple forms they use to generate the data the drives the application. Each of these forms provides the user with fields to enter or choose data matching the fields and restrictions defined in the Contenful models. This restricts what the user can enter and choose from removing the ability to enter or choose incorrect data. Fields that correspond to a url can only hold a valid url, fields that must be an image can only link to images, fields that must be content can only link to content, and so on. Contentful also provides the ability to upload images so content and pages that use images simply need to have those images uploaded and selected in Contentful. An example of how the model definition for the ArticleTemplate looks within Contentful is shown in Figure 6. An example of the corresponding data entry form Contentful creates for the model AtricleTemplate is shown in Figure 7. Contentful also has APIs and scripts available for migrating models and data from one space to another giving users the ability to easily copy OrgTrac's sample data from the OrgTrac project in Contentful over to their own organizations space.

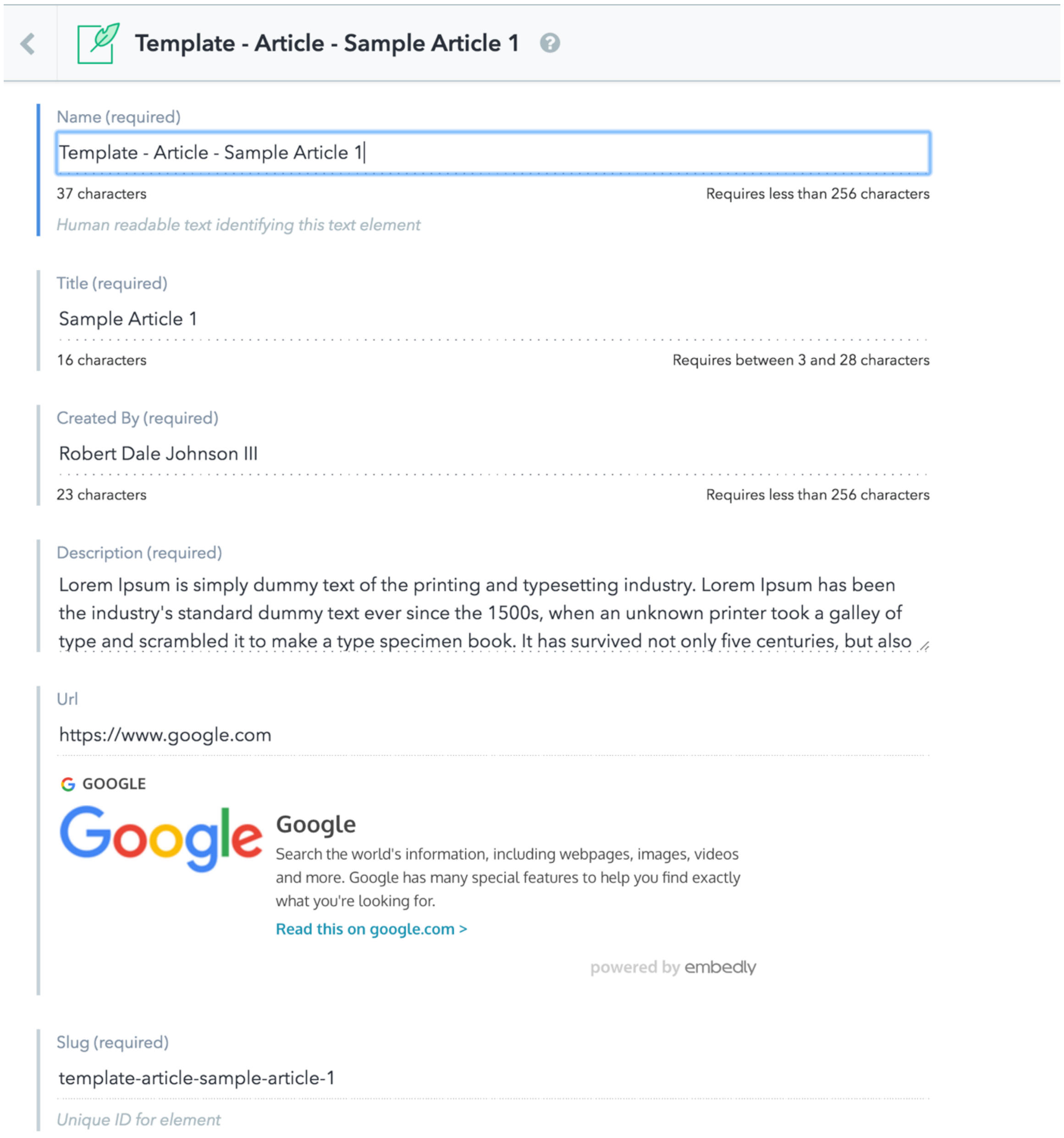


Template - Article Basic Article Template Edit

Fields (6) JSON preview

Name	Short text	Entry title	Settings
Title	Short text		Settings
Created By	Short text		Settings
Description	Long text		Settings
Url	Short text		Settings
Slug	Short text		Settings

Figure 6 - Contentful ArticleTemplate Model Definition



Template - Article - Sample Article 1

Name (required)

Template - Article - Sample Article 1

37 characters | Requires less than 256 characters

Human readable text identifying this text element

Title (required)

Sample Article 1

16 characters | Requires between 3 and 28 characters

Created By (required)

Robert Dale Johnson III

23 characters | Requires less than 256 characters

Description (required)

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also

Url

https://www.google.com

GOOGLE

Google

Search the world's information, including webpages, images, videos and more. Google has many special features to help you find exactly what you're looking for.

Read this on google.com >

powered by embedly

Slug (required)

template-article-sample-article-1

Unique ID for element

Figure 7 - Contentful ArticleTemplate Model Form

4.3 The Wiki

The Wiki ties all the different pieces of this project together walking the user through the different steps and processes needed in order to build their organization an application. The wiki handles walking the user through every step starting with downloading all the necessary software required including Android Studio, and GitHub Desktop in the Get Your Tools section. The wiki then walks the user through the steps of creating the necessary accounts needed in order to build and publish the application including setting up accounts for GitHub, Contentful, and Google Developer in the Using the Tools section. From here the wiki will walk the user through the many process needed to use all these tools to build the organization's application such as getting the source code and modifying it, generating the application binary, and publishing the app in the Using the Tools section.

The Get Your Tools section consists of linking the user to the resources to download the GtiHub Desktop and Android Studio applications. This section is simple since both applications have intuitive, well defined processes for downloading and installing the applications for the operating systems that they are available for. The wiki makes some general recommendations for the software and summarizes some of the process.

The Register for Online Tools sections consists of linking the user to the online tools needed and provides some insight into their usage. This section is also similarly simple since GitHub, the Google Play Store, and Contentful all also provide intuitive, well defined processes for registering for and creating accounts within their systems. The wiki makes some general recommendations for the tools and summarizes the process.

The Using the Tools section consists of the majority of the efforts involved in using the downloaded software and online tools to create the user's organization application. First the wiki walks through downloading the entire OrgTrac project from GitHub using GitHub Desktop. The user is then walked through the process of creating a space within Contentful that they will use for their organization. They are walked through the process of importing the OrgTrac Content Models and associated sample content from the OrgTrac sample project space within Contentful into the organizations space using the script provided within the project files downloaded from GitHub. The wiki continues with the process of opening the Android Studio project so the appropriate files and values can be modified to reflect the organization. Several files need to be updated to reflect the Contentful project, new organizations name and configuration values. Several other option changes such as updated strings, color changes and the creation of an application icon are also presented. The final steps of the process include generating a signature certificate and building the application binary so that it can be publish through the Google Play Store.

An additional sections with additional topics related to the project and modifying its data to match the user's organization as well as a few helpful facts and additional information is also included.

5. Project Results

To establish whether this project was a success, one must be able to confidently say that the goal was met. In the case of this project, the goal was to implement a resource that would allow a user to build an Android application for their organization at little cost, with minimal expertise, and to be able to drive that application from a dynamic resource. By following the wiki and using the resources provided with it a user can accomplish exactly that. A user with basic computer skills and less than $100 is able to use the wiki to create and publish an Android application to the Google Play Store that nearly anyone can download and use.

5.1 Analysis – Project Successes

There are several key milestone successes that contributed to the success of the overall project. The first and most foundational of these successes was defining the models for the content of the application. Defining the models in a way that was not only powerful and reusable but also intuitive and simple allowed for the success of the rest of the project since without those models, nothing else could have been built.

The second success was finding and mirroring the models in a system that would allow for dynamic delivery of the data to the application. Choosing a CMS that allowed for precise model definitions and easy to use data creation via forms was paramount to providing the user with a solution for delivering dynamic content to the application. Contentful was the perfect tool for the job due to its ease of use, low cost, and plethora of libraries available to developers.

The third success was designing and coding an application that would provide all the necessary features of an application through a generic implementation of models that would be defined and stored outside of the application but used to generate the entire structure and content of the application. This was accomplished with the OrgTrac project while keeping the modifications to change it to an entire new organization’s application as simple as changing a few values, within a couple files and to do so as simply as updating a word document.

The final major success was creating the wiki that pulled all the resources of the project together and ultimately walked the user through the process, step-by-step, in a way that was simple to find and simple to use since it is presented as a simple wiki website that the user can click through. Every step of the process is easy to find and simple to return to whenever necessary.

5.2 Analysis – Timing Problems

One problem that hindered several aspects of the project was the shortened time period to develop the entirety of this project. Several aspects of the project could have been done more precisely and even more intuitively had more time been available. Additional efforts to automate the process of updating the application could have been made if more time were available significantly easing the process for the user. An early goal of allowing the application to function without an internet connection by using a seed database was removed since it require quite a bit more effort in coding and documentation of the process for which time simply did not exist. However, despite this issue the project was still able to accomplish its goal without having to sacrifice any aspects of the project that would have hindered the user's ability to create their application

5.3 Analysis – Project Difficulties

The most difficult part of this project presented itself throughout the entire process of working on this project. Each piece from designing the models, to designing and implementing the application, to picking the dynamic delivery of the content, to writing the wiki all had to be able to be presented in a simple and intuitive way for any user. Tasks that typically take specific expertise, a large body of knowledge, vast funds, and numerous hours had to be designed, implemented, and presented to the user in a way that did not require any of these things.

By no means is any project easy or without difficulties but OrgTrac proved this point in many ways. Each design decision, each tool choice, each task that was left to the user to complete had to be made not only to accomplish the task at hand but had to be able to do the job as easily as possible. The wrong choice of tool, wrong definition of content, or simply ambiguous step in the wiki could have easily cause the failure of this project. However, enough decisions and choices were made correctly that the difficulties of creating this project were mostly avoided allowing the goals of this project to be accomplished.

6. Conclusion

The desire to provide a resource giving groups like a high school football team, chess club, cub scout den, robotics club, or any other group who might wish to create a mobile application but on their own would never have the ability to do so was what lead to the creation of this project. I wanted to give anyone who wished to share their passion with others the ability to do so without having to spend tons of money or use a system that would take away from them being able to freely present their content to their audience.

I had to create a resource that would in effect do “all the heavy lifting” related to developing an Android application allowing the user the ability to spend as little time and money possible on creating the application so they could focus on their passion and make the content they want their audience to see. Prior to the final approach of the project I iterated over possible model designs, possible tools, and possible resources eventually falling into the tools and resources that would ultimately allow this project to succeed. With plenty of trial and error along with a little dumb luck that a company I was working with adopted Contentful as a CMS to drive their products I was able to finally create the resource I’ve wanted to for the last 7 years. Having been able to finally accomplish these goals and show the process of creating it and why I chose to do so, what remains is making it even better and more universally available to the users and their audiences.

6.1 Future Works

As with the completion of any project, whether it be in industry or in academia, the true success of a project lies in its use and future, whether that project is able to evolve and continue to provide its purpose to its users. This project only scratches the service of what is possible and can provide so much more in so many ways making the process easier and faster while providing even more features and abilities to the user. Extending its abilities and resources in ways that make it more useful to more groups across more platforms.

Adding additional features to this project evolving its abilities and the audience it can reach is the most appealing. Things that can appear very simple at face value such as adding additional templates and elements can broaden the types of organization that might find using OrgTrac beneficial. Adding templates for accepting payments or donations can open OrgTrac to groups looking to sell their products, start a charity, or even collect dues for a club. Adding support for different CMSs could allow groups already tied to, or familiar with, other CMSs to use those CMSs to drive their mobile application using OrgTrac. Adding support for additional platforms other than Android such as iOS or the web can open the doors to bigger audiences. Adding additional sample projects in Contentful showing how different organizations might define their data could give concrete working examples to organizations to follow.

Another idea to make the project better is to supplement the existing project making it even easier to use. Visual aids such as detailed screenshots and even videos recordings of the steps defined in the wiki can be made allowing users to both read and watch the necessary steps to complete creating their application. Other efforts can also be made to automate some of the manual steps the users must do such as providing an application that prompts the user for the values that need to be updated in the OrgTrac project and automatically updating them for the user. With the right resources it might even be possible to automate all the processes up to publishing the application making the goal of this project accessible to even more people with even less effort.

References

[1] How Much Does it Cost to Make An App in 2017?. *Codementorx Web site.* [Online] [Cited: Sep 17, 2018.] . https://www.codementor.io/blog/how-much-does-it-cost-to-ma ke-an-app-in-2017-1nqj6ehste

[2] git --distributed-even-if-your-workflow-isnt. *Git SCM site.* [Online] [Cited: Oct 02, 2018.]. https://git-scm.com/

[3] Bitbucket – Built for professional teams. *Atlassian Bitbucket site.* [Online] [Cited: Oct 02, 2018.]. https://bitbucket.org/

[4] GitHub - Built for developers. *GitHub site.* [Online] [Cited: Oct 02, 2018.]. https://github.com/

[5] Android is for everyone. *Android Web site.* [Online] [Cited: Oct 02, 2018.]. https://www.android.com/everyone/

[6] Ship your digital products faster. *Contentful Web site.* [Online] [Cited: Oct 02, 2018.]. https://www.contentful.com/why-contentful/

[7] Headless CMS. *Wikipedia Web site.* [Online] [Cited: Oct 04, 2018.]. https://en.wikipedia.org/wiki/Headless_CMS

[8] Support Library. *Android Developer site.* [Online] [Cited: Oct 05, 2018.]. https://developer.android.com/topic/libraries/support-library/

[9] Developer Docs. *Contentful site.* [Online] [Cited: Oct 05, 2018.]. https://www.contentful.com/developers/docs/

[10] vault - Contentful Offline Persistence for Android. *GitHub site.* [Online] [Cited: Oct 05, 2018.]. https://github.com/contentful/vault

[11] KAndroid – Kotlin library for Android. *GitHub site*. [Online] [Cited: Oct 08, 2018]. https://github.com/pawegio/KAndroid

[12] Anko - Pleasant Android application development. *GitHub site*. [Online] [Cited: Oct 08, 2018]. https://github.com/Kotlin/anko

[13] RxJava: Reactive Extensions for the JVM. *GitHub site*. [Online] [Cited: Oct 08. 2018]. https://github.com/ReactiveX/RxJava

[14] RxAndroid - RxJava bindings for Android. *GitHub site*. [Online] [Cited: Oct 08, 2018]. https://github.com/ReactiveX/RxAndroid

[15] Dagger 2. *Google GitHub site*. [Online] [Cited: Oct 08. 2018]. https://google.github.io/dagger/

[16] Timber - A logger with a small, extensible API which provides utility on top of Android's normal Log class. *GitHub site*. [Online] [Cited: Oct 08. 2018] https://github.com/JakeWharton/timber

[17] Picasso - A powerful image downloading and caching library for Android. *Square GitHub site.* [Online] [Cited: Oct 08, 2018] http://square.github.io/picasso/

[18] The new native. *GitHub Desktop site*. [Online] [Cited: Oct 09, 2018] https://desktop.github.com/

Appendix A – LoadingActivity.kt Source Code

```
package edu.csun.orgtrac.activity

import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import com.pawegio.kandroid.toast
import com.squareup.picasso.Picasso
import edu.csun.orgtrac.DataManager
import edu.csun.orgtrac.R
import edu.csun.orgtrac.component
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.schedulers.Schedulers
import kotlinx.android.synthetic.main.activity_loading.loading_image
import kotlinx.android.synthetic.main.activity_loading.loading_message
import java.util.concurrent.TimeUnit
import javax.inject.Inject

class LoadingActivity : AppCompatActivity() {
  @Inject lateinit var dataManager: DataManager
  @Inject lateinit var picasso: Picasso

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_loading)
    component.inject(this)
  }

  override fun onResume() {
    super.onResume()
    updateLoadingPage()
    dataManager.translateData()
```

```
            .timeout(10, TimeUnit.SECONDS)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(
                { myData ->
                    startActivity(LandingActivity.newIntent(this))
                    finish()
                },
                { throwable ->
                    toast(throwable.message.orEmpty())
                }
            )
    }

    private fun updateLoadingPage() {
        dataManager.loadingPage?.let {
            title = it.title
            loading_message.text = it.message
            picasso
                .load(it.image)
                .into(loading_image)
        }
    }
}
```

Appendix B – LandingActivity.kt Source Code

```
package edu.csun.orgtrac.activity

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.support.v4.content.ContextCompat
import android.support.v7.app.AppCompatActivity
import com.pawegio.kandroid.toast
import edu.csun.orgtrac.DataManager
import edu.csun.orgtrac.OrgTrac
import edu.csun.orgtrac.R
import edu.csun.orgtrac.component
import edu.csun.orgtrac.launchUrl
import kotlinx.android.synthetic.main.activity_landing.landing_content
import kotlinx.android.synthetic.main.activity_landing.toolbar
import kotlinx.android.synthetic.main.activity_landing_item.view.landing_card_image
import kotlinx.android.synthetic.main.activity_landing_item.view.landing_card_title
import javax.inject.Inject

class LandingActivity : AppCompatActivity() {
   @Inject lateinit var dataManager: DataManager

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_landing)
      setSupportActionBar(toolbar)
      component.inject(this)

      title = dataManager.landingPage?.title
```

```
        landing_content.removeAllViews()
        dataManager.landingPage?.pages?.forEach { mainPage ->
            val itemView = layoutInflater.inflate(R.layout.activity_landing_item,
landing_content, false)

            itemView.landing_card_title.text = mainPage.title
            itemView.landing_card_image.background = ContextCompat.getDrawable(this,
R.mipmap.content)

            mainPage.content?.let { content ->
                if (content.size == 1) {
                    when (content.first()) {
                        is OrgTrac.Model.Content.Template.Web -> {
                            itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.internet)
                        }
                        is OrgTrac.Model.Content.Template.Profile -> {
                            itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.profile)
                        }
                        is OrgTrac.Model.Content.Template.Location -> {
                            itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.location)
                        }
                        is OrgTrac.Model.Content.Template.Article -> {
                            itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.article)
                        }
                        is OrgTrac.Model.Content.Template.ContactInfo -> {
                            itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.contact)
                        }
                        is OrgTrac.Model.Content.Element.Youtube -> {
```

```
                    itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.youtube)
                  }
                  is OrgTrac.Model.Content.Element.Link -> {
                    itemView.landing_card_image.background =
ContextCompat.getDrawable(this, R.mipmap.internet)
                  }
                }
              }
            }

            itemView.setOnClickListener {
              mainPage.content?.let { content ->
                if (content.isNotEmpty()) {
                  if (content.size == 1 && content.first() is
OrgTrac.Model.Content.Template.Web) {
                    launchUrl((content.first() as
OrgTrac.Model.Content.Template.Web).url.orEmpty())
                  } else {
                    startActivity(MainActivity.newIntent(this, mainPage.slug.orEmpty()))
                  }
                } else {
                  toast("Sorry, this screen in not defined yet.")
                }
              }
            }

            landing_content.addView(itemView)
          }
        }

        companion object {
```

```
        fun newIntent(context: Context) = Intent(context, LandingActivity::class.java)
    }
}
```

Appendix C – MainActivity.kt Source Code

```
package edu.csun.orgtrac.activity


import android.content.Context
import android.content.Intent
import android.graphics.Typeface
import android.net.Uri
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.view.View.VISIBLE
import com.squareup.picasso.Picasso
import edu.csun.orgtrac.DataManager
import edu.csun.orgtrac.OrgTrac
import edu.csun.orgtrac.R
import edu.csun.orgtrac.component
import edu.csun.orgtrac.launchUrl
import kotlinx.android.synthetic.main.activity_main.main_content
import kotlinx.android.synthetic.main.activity_main.toolbar
import
kotlinx.android.synthetic.main.activity_main_article_template_item.view.article_created
_by
import
kotlinx.android.synthetic.main.activity_main_article_template_item.view.article_descript
ion
import
kotlinx.android.synthetic.main.activity_main_article_template_item.view.article_title
import
kotlinx.android.synthetic.main.activity_main_article_template_item.view.article_url
import
kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_i
nfo_description
import
kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_i
nfo_email
```

```
import kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_info_facebook
import kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_info_facebook_text
import kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_info_title
import kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_info_twitter
import kotlinx.android.synthetic.main.activity_main_contact_info_template_item.view.contact_info_twitter_text
import kotlinx.android.synthetic.main.activity_main_image_element_item.view.image_image
import kotlinx.android.synthetic.main.activity_main_link_element_item.view.link_title
import kotlinx.android.synthetic.main.activity_main_location_template_item.view.location_address
import kotlinx.android.synthetic.main.activity_main_location_template_item.view.location_image
import kotlinx.android.synthetic.main.activity_main_location_template_item.view.location_title
import kotlinx.android.synthetic.main.activity_main_profile_template_item.view.profile_description
import kotlinx.android.synthetic.main.activity_main_profile_template_item.view.profile_image
import kotlinx.android.synthetic.main.activity_main_profile_template_item.view.profile_title
import kotlinx.android.synthetic.main.activity_main_text_element_item.view.text_title
import kotlinx.android.synthetic.main.activity_main_web_template_item.view.web_title
```

```
import
kotlinx.android.synthetic.main.activity_main_youtube_video_element_item.view.youtub
e_video_title
import javax.inject.Inject

class MainActivity : AppCompatActivity() {
   @Inject lateinit var dataManager: DataManager
   @Inject lateinit var picasso: Picasso

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      setSupportActionBar(toolbar)
      supportActionBar?.setDisplayHomeAsUpEnabled(true)

      component.inject(this)
      val page = dataManager.mainPages[intent.getStringExtra(EXTRA_SLUG)]

      title = page?.title

      main_content.removeAllViews()
      page?.content?.forEach { content ->
         when (content) {
            is OrgTrac.Model.Content.Template.Web -> {
               val itemView =
layoutInflater.inflate(R.layout.activity_main_web_template_item, main_content, false)

               itemView.web_title.text = content.title
               itemView.setOnClickListener {
                  launchUrl(content.url.orEmpty())
               }
```

```
                main_content.addView(itemView)
            }
            is OrgTrac.Model.Content.Template.Profile -> {
                val itemView =
                    layoutInflater.inflate(R.layout.activity_main_profile_template_item,
main_content, false)

                itemView.profile_title.text = content.title
                itemView.profile_description.text = content.description
                picasso
                    .load(content.image)
                    .into(itemView.profile_image)
                itemView.setOnClickListener {
                    // At this time no actions are defined
                }

                main_content.addView(itemView)
            }
            is OrgTrac.Model.Content.Template.Location -> {
                val itemView =
                    layoutInflater.inflate(R.layout.activity_main_location_template_item,
main_content, false)

                itemView.location_title.text = content.title
                itemView.location_address.text = content.address
                itemView.setOnClickListener {
                    // At this time no actions are defined
                }

                if (content.latitude != null && content.longitude != null) {
                  val gmmIntentUri =
Uri.parse("geo:${content.latitude},${content.longitude}?q=" +
Uri.encode(content.address))
                  val mapIntent = Intent(Intent.ACTION_VIEW, gmmIntentUri)
                  mapIntent.setPackage("com.google.android.apps.maps")
                  if (mapIntent.resolveActivity(getPackageManager()) != null) {
                     itemView.location_image.visibility = VISIBLE
                     itemView.location_image.setOnClickListener {
                        startActivity(mapIntent)
                     }
                  }
               }

               main_content.addView(itemView)
            }
            is OrgTrac.Model.Content.Template.Article -> {
               val itemView =
                  layoutInflater.inflate(R.layout.activity_main_article_template_item,
main_content, false)

               itemView.article_title.text = content.title
               itemView.article_created_by.text = content.createdBy
               itemView.article_description.text = content.description
               itemView.article_url.text = content.url
               itemView.setOnClickListener {
                  // At this time no actions are defined
               }

               content.url?.let { url ->
                  itemView.article_url.visibility = VISIBLE
                  itemView.article_url.setOnClickListener {
                     launchUrl(content.url.orEmpty())
```

```
                    }
                }

                main_content.addView(itemView)
            }
            is OrgTrac.Model.Content.Template.ContactInfo -> {
                val itemView =
                    layoutInflater.inflate(R.layout.activity_main_contact_info_template_item,
main_content, false)

                itemView.contact_info_title.text = content.title
                itemView.contact_info_email.text = content.email
                itemView.contact_info_description.text = content.description

                itemView.contact_info_email.setOnClickListener {
                    val intent = Intent(Intent.ACTION_SEND).apply {
                        type = "*/*"
                        putExtra(Intent.EXTRA_EMAIL, arrayOf(content.email))
                        putExtra(Intent.EXTRA_SUBJECT, content.title)
                    }
                    if (intent.resolveActivity(packageManager) != null) {
                        startActivity(intent)
                    }
                }

                content.facebookTag?.let { facebookTag ->
                    itemView.contact_info_facebook.visibility = VISIBLE
                    itemView.contact_info_facebook_text.text = "@$facebookTag"
                    itemView.contact_info_facebook.setOnClickListener {
                        // Maybe launch in the future
                    }
                }

                content.twitterTag?.let { twitterTag ->
                    itemView.contact_info_twitter.visibility = VISIBLE
                    itemView.contact_info_twitter_text.text = "@$twitterTag"
                    itemView.contact_info_twitter.setOnClickListener {
                        // Maybe launch in the future
                    }
                }

                main_content.addView(itemView)
            }
            is OrgTrac.Model.Content.Element.Text -> {
                val itemView =
layoutInflater.inflate(R.layout.activity_main_text_element_item, main_content, false)

                itemView.text_title.text = content.text
                when (content.fontStyle) {
                    "Italic" -> {
                        itemView.text_title.setTypeface(itemView.text_title.getTypeface(),
Typeface.ITALIC)
                    }
                    "Bold" -> {
                        itemView.text_title.setTypeface(itemView.text_title.getTypeface(),
Typeface.BOLD)
                    }
                    else -> {
                        itemView.text_title.setTypeface(itemView.text_title.getTypeface(),
Typeface.NORMAL)
                    }
                }
```

```
          main_content.addView(itemView)
        }
        is OrgTrac.Model.Content.Element.Image -> {
          val itemView =
            layoutInflater.inflate(R.layout.activity_main_image_element_item,
main_content, false)

          itemView.image_image.contentDescription = content.description
          picasso
            .load(content.image)
            .into(itemView.image_image)
          itemView.setOnClickListener {
            // At this time no actions are defined
          }

          main_content.addView(itemView)
        }
        is OrgTrac.Model.Content.Element.Youtube -> {
          val itemView =

layoutInflater.inflate(R.layout.activity_main_youtube_video_element_item,
main_content, false)

          itemView.youtube_video_title.text = content.title
          content.url?.let { url ->
            itemView.setOnClickListener {
              launchUrl(content.url.orEmpty())
            }
          }

          main_content.addView(itemView)
        }
```

```
                is OrgTrac.Model.Content.Element.Link -> {
                    val itemView =
layoutInflater.inflate(R.layout.activity_main_link_element_item, main_content, false)

                    itemView.link_title.text = content.displayText
                    content.url?.let { url ->
                        itemView.setOnClickListener {
                            launchUrl(content.url.orEmpty())
                        }
                    }

                    main_content.addView(itemView)
                }
            }
        }
    }

    companion object {
        private const val EXTRA_SLUG = "extra:slug"

        fun newIntent(context: Context, slug: String): Intent {
            val intent = Intent(context, MainActivity::class.java)
            intent.putExtra(EXTRA_SLUG, slug)

            return intent
        }
    }
}
```

Appendix D – _contentful.kt Source Code

```
package edu.csun.orgtrac

import com.contentful.vault.Asset
import com.contentful.vault.ContentType
import com.contentful.vault.Field
import com.contentful.vault.Resource
import com.contentful.vault.Space

@Space(
    value = "re77qk2q3c1m",
    models = [
        Contentful.Model.LoadingPage::class,
        Contentful.Model.LandingPage::class,
        Contentful.Model.MainPage::class,
        Contentful.Model.WebTemplate::class,
        Contentful.Model.ProfileTemplate::class,
        Contentful.Model.LocationTemplate::class,
        Contentful.Model.ArticleTemplate::class,
        Contentful.Model.ContactInfoTemplate::class,
        Contentful.Model.TextElement::class,
        Contentful.Model.ImageElement::class,
        Contentful.Model.YoutubeElement::class,
        Contentful.Model.LinkElement::class,
        Contentful.Model.LinkReference::class
    ],
    locales = ["en-US"],
    dbVersion = 1
)
class Space
class Contentful{
```

```
class Model {
    @ContentType("pageLoading")
    data class LoadingPage(
        @Field("name") @JvmField var name: String? = null,
        @Field @JvmField var title: String? = null,
        @Field @JvmField var image: Asset? = null,
        @Field @JvmField var message: String? = null,
        @Field @JvmField var slug: String? = null
    ) : Resource()

    @ContentType("pageLanding")
    data class LandingPage(
        @Field @JvmField var name: String? = null,
        @Field @JvmField var title: String? = null,
        @Field @JvmField var pages: List<MainPage>? = null,
        @Field @JvmField var slug: String? = null
    ) : Resource()

    @ContentType("pageMain")
    data class MainPage(
        @Field @JvmField var name: String? = null,
        @Field @JvmField var title: String? = null,
        @Field @JvmField var content: List<LinkReference>? = null,
        @Field @JvmField var slug: String? = null
    ) : Resource()

    @ContentType("templateWeb")
    data class WebTemplate(
        @Field @JvmField var name: String? = null,
        @Field @JvmField var title: String? = null,
        @Field @JvmField var url: String? = null,
    @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("templateProfile")
data class ProfileTemplate(
    @Field @JvmField var name: String? = null,
    @Field @JvmField var image: Asset? = null,
    @Field @JvmField var title: String? = null,
    @Field @JvmField var description: String? = null,
    @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("templateLocation")
data class LocationTemplate(
    @Field @JvmField var name: String? = null,
    @Field @JvmField var title: String? = null,
    @Field @JvmField var address: String? = null,
    @Field @JvmField var latitude: Double? = null,
    @Field @JvmField var longitude: Double? = null,
    @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("templateArticle")
data class ArticleTemplate(
    @Field @JvmField var name: String? = null,
    @Field @JvmField var title: String? = null,
    @Field @JvmField var createdBy: String? = null,
    @Field @JvmField var description: String? = null,
    @Field @JvmField var url: String? = null,
    @Field @JvmField var slug: String? = null
) : Resource()
```

```
@ContentType("templateContactInfo")
data class ContactInfoTemplate(
   @Field @JvmField var name: String? = null,
   @Field @JvmField var title: String? = null,
   @Field @JvmField var email: String? = null,
   @Field @JvmField var description: String? = null,
   @Field @JvmField var facebookTag: String? = null,
   @Field @JvmField var twitterTag: String? = null,
   @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("elementText")
data class TextElement(
   @Field @JvmField var name: String? = null,
   @Field @JvmField var text: String? = null,
   @Field @JvmField var fontStyle: String? = null,
   @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("elementImage")
data class ImageElement(
   @Field @JvmField var name: String? = null,
   @Field @JvmField var image: Asset? = null,
   @Field @JvmField var description: String? = null,
   @Field @JvmField var slug: String? = null
) : Resource()

@ContentType("elementYouTubeVideo")
data class YoutubeElement(
   @Field @JvmField var name: String? = null,
            @Field @JvmField var title: String? = null,
            @Field @JvmField var url: String? = null,
            @Field @JvmField var slug: String? = null
        ) : Resource()

        @ContentType("elementLink")
        data class LinkElement(
            @Field @JvmField var name: String? = null,
            @Field @JvmField var displayText: String? = null,
            @Field @JvmField var url: String? = null,
            @Field @JvmField var slug: String? = null
        ) : Resource()

        @ContentType("linkReference")
        data class LinkReference(
            @Field @JvmField var name: String? = null,
            @Field @JvmField var linkedType: String? = null,
            @Field @JvmField var linkedSlug: String? = null,
            @Field @JvmField var slug: String? = null
        ) : Resource()
    }
}
```

Appendix E – _orgtrac.kt Source Code

```
package edu.csun.orgtrac

import timber.log.Timber

class OrgTrac {
    class Model {
        data class LoadingPage(
            var title: String? = null,
            var image: String? = null,
            var message: String? = null
        ) {
            companion object {
                fun build(loadingPage: Contentful.Model.LoadingPage?): LoadingPage? {
                    if (loadingPage != null) {
                        return LoadingPage(
                            loadingPage.title,
                            loadingPage.image?.url(),
                            loadingPage.message
                        )
                    }

                    return null
                }
            }
        }

        data class LandingPage(
            var title: String? = null,
            var pages: List<MainPage>? = null
        ) {
```

```
        companion object {
            fun build(
                landingPage: Contentful.Model.LandingPage?,
                webTemplates: HashMap<String, OrgTrac.Model.Content.Template.Web>,
                profileTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Profile>,
                locationTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Location>,
                articleTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Article>,
                contactInfoTemplates: HashMap<String,
OrgTrac.Model.Content.Template.ContactInfo>,
                textElements: HashMap<String, OrgTrac.Model.Content.Element.Text>,
                imageElements: HashMap<String, OrgTrac.Model.Content.Element.Image>,
                youtubeVideoElements: HashMap<String,
OrgTrac.Model.Content.Element.Youtube>,
                linkElements: HashMap<String, OrgTrac.Model.Content.Element.Link>
            ): LandingPage? {
                if (landingPage != null) {
                    val pages = ArrayList<MainPage>()
                    landingPage.pages?.forEach { mainPage ->
                        val page = MainPage.build(
                            mainPage,
                            webTemplates,
                            profileTemplates,
                            locationTemplates,
                            articleTemplates,
                            contactInfoTemplates,
                            textElements,
                            imageElements,
                            youtubeVideoElements,
                            linkElements
                        )
                        if (page != null) {
                            pages.add(page)
                        }
                    }

                    return LandingPage(landingPage.title, pages)
                }

                return null
            }
        }
    }

    data class MainPage(
        var title: String? = null,
        var content: List<Content>? = null,
        var slug: String? = null
    ) {
        companion object {
            fun build(
                mainPage: Contentful.Model.MainPage?,
                webTemplates: HashMap<String, OrgTrac.Model.Content.Template.Web>,
                profileTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Profile>,
                locationTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Location>,
                articleTemplates: HashMap<String,
OrgTrac.Model.Content.Template.Article>,
                contactInfoTemplates: HashMap<String,
OrgTrac.Model.Content.Template.ContactInfo>,
                textElements: HashMap<String, OrgTrac.Model.Content.Element.Text>,
              imageElements: HashMap<String, OrgTrac.Model.Content.Element.Image>,
              youtubeVideoElements: HashMap<String,
OrgTrac.Model.Content.Element.Youtube>,
              linkElements: HashMap<String, OrgTrac.Model.Content.Element.Link>
            ): MainPage? {
              if (mainPage != null) {
                val contents = ArrayList<OrgTrac.Model.Content>()

                mainPage.content?.forEach { linkReference ->
                  val content: Content? = when (linkReference.linkedType) {
                    "Template - Web" -> webTemplates[linkReference.linkedSlug]
                    "Template - Profile" -> profileTemplates[linkReference.linkedSlug]
                    "Template - Location" ->
locationTemplates[linkReference.linkedSlug]
                    "Template - Article" -> articleTemplates[linkReference.linkedSlug]
                    "Template - Contact Info" ->
contactInfoTemplates[linkReference.linkedSlug]
                    "Element - Text" -> textElements[linkReference.linkedSlug]
                    "Element - Image" -> imageElements[linkReference.linkedSlug]
                    "Element - YouTube Video" ->
youtubeVideoElements[linkReference.linkedSlug]
                    "Element - Link" -> linkElements[linkReference.linkedSlug]
                    else -> {
                      null
                    }
                  }

                  if (content == null) {
                    Timber.d("Error Finding (${linkReference.linkedSlug}) in
(${linkReference.linkedType})")
                  }
```

```
                    content?.let {
                        contents.add(content)
                    }
                }

                return MainPage(
                    mainPage.title,
                    contents,
                    mainPage.slug
                )
            }

            return null
        }
    }
}

sealed class Content {
    sealed class Template : Content() {
        data class Web(
            var title: String? = null,
            var url: String? = null
        ) : Template() {
            companion object {
                fun build(template: Contentful.Model.WebTemplate?): Web? {
                    if (template != null) {
                        return Web(
                            template.title,
                            template.url
                        )
                    }
            return null
        }
    }
}

data class Profile(
    var image: String? = null,
    var title: String? = null,
    var description: String? = null
) : Template() {
    companion object {
        fun build(template: Contentful.Model.ProfileTemplate?): Profile? {
            if (template != null) {
                return Profile(
                    template.image?.url(),
                    template.title,
                    template.description
                )
            }

            return null
        }
    }
}

data class Location(
    var title: String? = null,
    var address: String? = null,
    var latitude: Double? = null,
    var longitude: Double? = null
```

```
) : Template() {
    companion object {
        fun build(template: Contentful.Model.LocationTemplate?): Location? {
            if (template != null) {
                return Location(
                    template.title,
                    template.address,
                    template.latitude,
                    template.longitude
                )
            }

            return null
        }
    }
}

data class Article(
    var title: String? = null,
    var createdBy: String? = null,
    var description: String? = null,
    var url: String? = null
) : Template() {
    companion object {
        fun build(template: Contentful.Model.ArticleTemplate?): Article? {
            if (template != null) {
                return Article(
                    template.title,
                    template.createdBy,
                    template.description,
                    template.url
                )
              }

              return null
            }
          }
        }

        data class ContactInfo(
          var title: String? = null,
          var email: String? = null,
          var description: String? = null,
          var facebookTag: String? = null,
          var twitterTag: String? = null
        ) : Template() {
          companion object {
            fun build(template: Contentful.Model.ContactInfoTemplate?):
ContactInfo? {
              if (template != null) {
                return ContactInfo(
                  template.title,
                  template.email,
                  template.description,
                  template.facebookTag,
                  template.twitterTag
                )
              }

              return null
            }
          }
```

```
    }
}

sealed class Element : Content() {
    data class Text(
        var text: String? = null,
        var fontStyle: String? = null
    ) : Element() {
        companion object {
            fun build(element: Contentful.Model.TextElement?): Text? {
                if (element != null) {
                    return Text(
                        element.text,
                        element.fontStyle
                    )
                }

                return null
            }
        }
    }

    data class Image(
        var image: String? = null,
        var description: String? = null
    ) : Element() {
        companion object {
            fun build(element: Contentful.Model.ImageElement?): Image? {
                if (element != null) {
                    return Image(
                        element.image?.url(),
                element.description
            )
        }

        return null
      }
    }
}

data class Youtube(
    var title: String? = null,
    var url: String? = null
) : Element() {
    companion object {
        fun build(element: Contentful.Model.YoutubeElement?): Youtube? {
            if (element != null) {
                return Youtube(
                    element.title,
                    element.url
                )
            }

            return null
        }
    }
}

data class Link(
    var displayText: String? = null,
    var url: String? = null
) : Element() {
```

```
                    companion object {
                        fun build(element: Contentful.Model.LinkElement?): Link? {
                            if (element != null) {
                                return Link(
                                    element.displayText,
                                    element.url
                                )
                            }

                            return null
                        }
                    }
                }
            }
        }
    }
}
```

Appendix F – DataManager.kt Source Code

```
package edu.csun.orgtrac

import com.contentful.vault.SyncConfig
import com.contentful.vault.SyncResult
import com.contentful.vault.Vault
import io.reactivex.Single

class DataManager(val vault: Vault, val syncConfig: SyncConfig) {
   var webTemplates = HashMap<String, OrgTrac.Model.Content.Template.Web>()
   var profileTemplates = HashMap<String, OrgTrac.Model.Content.Template.Profile>()
   var locationTemplates = HashMap<String,
OrgTrac.Model.Content.Template.Location>()
   var articleTemplates = HashMap<String, OrgTrac.Model.Content.Template.Article>()
   var contactInfoTemplates = HashMap<String,
OrgTrac.Model.Content.Template.ContactInfo>()
   var textElements = HashMap<String, OrgTrac.Model.Content.Element.Text>()
   var imageElements = HashMap<String, OrgTrac.Model.Content.Element.Image>()
   var youtubeVideoElements = HashMap<String,
OrgTrac.Model.Content.Element.Youtube>()
   var linkElements = HashMap<String, OrgTrac.Model.Content.Element.Link>()
   var loadingPage: OrgTrac.Model.LoadingPage? = null
   var landingPage: OrgTrac.Model.LandingPage? = null
   var mainPages: HashMap<String, OrgTrac.Model.MainPage> = HashMap()

   fun translateData(): Single<SyncResult> {
      return Single.fromCallable {
         vault.requestSync(syncConfig)
         val syncResult = Vault.observeSyncResults().blockingFirst()

         vault.fetch(Contentful.Model.WebTemplate::class.java).all().forEach { temp ->
            OrgTrac.Model.Content.Template.Web.build(temp)?.let {
```

```
            webTemplates[temp.slug.orEmpty()] = it
        }
    }

    vault.fetch(Contentful.Model.ProfileTemplate::class.java).all().forEach { temp ->
        OrgTrac.Model.Content.Template.Profile.build(temp)?.let {
            profileTemplates[temp.slug.orEmpty()] = it
        }
    }

    vault.fetch(Contentful.Model.LocationTemplate::class.java).all().forEach { temp -
>
        OrgTrac.Model.Content.Template.Location.build(temp)?.let {
            locationTemplates[temp.slug.orEmpty()] = it
        }
    }

    vault.fetch(Contentful.Model.ArticleTemplate::class.java).all().forEach { temp ->
        OrgTrac.Model.Content.Template.Article.build(temp)?.let {
            articleTemplates[temp.slug.orEmpty()] = it
        }
    }

    vault.fetch(Contentful.Model.ContactInfoTemplate::class.java).all().forEach {
temp ->
        OrgTrac.Model.Content.Template.ContactInfo.build(temp)?.let {
            contactInfoTemplates[temp.slug.orEmpty()] = it
        }
    }

    vault.fetch(Contentful.Model.TextElement::class.java).all().forEach { ele ->
```

```
            OrgTrac.Model.Content.Element.Text.build(ele)?.let {
                textElements[ele.slug.orEmpty()] = it
            }
        }

        vault.fetch(Contentful.Model.ImageElement::class.java).all().forEach { ele ->
            OrgTrac.Model.Content.Element.Image.build(ele)?.let {
                imageElements[ele.slug.orEmpty()] = it
            }
        }

        vault.fetch(Contentful.Model.YoutubeElement::class.java).all().forEach { ele ->
            OrgTrac.Model.Content.Element.Youtube.build(ele)?.let {
                youtubeVideoElements[ele.slug.orEmpty()] = it
            }
        }

        vault.fetch(Contentful.Model.LinkElement::class.java).all().forEach { ele ->
            OrgTrac.Model.Content.Element.Link.build(ele)?.let {
                linkElements[ele.slug.orEmpty()] = it
            }
        }

        loadingPage =
OrgTrac.Model.LoadingPage.build(vault.fetch(Contentful.Model.LoadingPage::class.jav
a).first())
        landingPage = OrgTrac.Model.LandingPage.build(
            vault.fetch(Contentful.Model.LandingPage::class.java).first(),
            webTemplates,
            profileTemplates,
            locationTemplates,
                articleTemplates,
                contactInfoTemplates,
                textElements,
                imageElements,
                youtubeVideoElements,
                linkElements
            )

            vault.fetch(Contentful.Model.MainPage::class.java).all().forEach { page ->
                OrgTrac.Model.MainPage.build(
                    page,
                    webTemplates,
                    profileTemplates,
                    locationTemplates,
                    articleTemplates,
                    contactInfoTemplates,
                    textElements,
                    imageElements,
                    youtubeVideoElements,
                    linkElements
                )?.let {
                    mainPages[page.slug.orEmpty()] = it
                }
            }

            syncResult
        }
    }
}
```

Appendix G – AppModule.kt Source Code

```
package edu.csun.orgtrac.injection

import android.app.Application
import android.content.Context
import com.contentful.java.cda.CDAClient
import com.contentful.vault.SyncConfig
import com.contentful.vault.Vault
import com.squareup.picasso.OkHttp3Downloader
import com.squareup.picasso.Picasso
import dagger.Module
import dagger.Provides
import edu.csun.orgtrac.DataManager
import edu.csun.orgtrac.R
import edu.csun.orgtrac.Space
import okhttp3.OkHttpClient
import java.util.concurrent.TimeUnit
import javax.inject.Singleton

@Module
class AppModule(private val application: Application) {
   @Provides
   internal fun provideContext(): Context {
      return application
   }

   @Provides
   @Singleton
   fun provideVault(context: Context): Vault {
      return Vault.with(context, Space::class.java)
   }
```

```
@Provides
@Singleton
fun provideCDAClient(context: Context): CDAClient {
   return CDAClient.builder()
      .setToken(context.getString(R.string.cf_cda_token))
      .setSpace(context.getString(R.string.cf_space_id))
      .build()
}

@Provides
@Singleton
fun provideSync(cdaClient: CDAClient): SyncConfig {
   return SyncConfig.builder()
      .setClient(cdaClient)
      .setInvalidate(true)
      .build()
}

@Provides
@Singleton
fun provideDataManager(vault: Vault, syncConfig: SyncConfig): DataManager {
   return DataManager(vault, syncConfig)
}

@Provides
@Singleton
fun providePicasso(context: Context): Picasso {
   return Picasso.Builder(context)
      .downloader(
         OkHttp3Downloader(
                OkHttpClient.Builder()
                    .connectTimeout(10, TimeUnit.SECONDS)
                    .writeTimeout(10, TimeUnit.SECONDS)
                    .readTimeout(10, TimeUnit.SECONDS)
                    .build()
            )
        )
        .build()
    }
}
```

Appendix H – Layout Resource Source Code

activity_loading.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="@dimen/activity_horizontal_margin"
    android:layout_marginTop="@dimen/activity_vertical_margin"
    android:layout_marginRight="@dimen/activity_horizontal_margin"
    android:layout_marginBottom="@dimen/activity_vertical_margin"
    tools:context=".activity.LoadingActivity">

    <LinearLayout
        android:id="@+id/loading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/loading_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="centerInside"
            android:layout_margin="@dimen/image_margin"/>

        <ProgressBar
            android:id="@+id/loading_progress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView
            android:id="@+id/loading_message"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="@string/loading"/>

    </LinearLayout>
</RelativeLayout>
```

activity_landing.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:orientation="vertical"
    tools:context=".activity.LandingActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>
```

```
 <android.support.v4.widget.NestedScrollView
     xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     app:layout_behavior="@string/appbar_scrolling_view_behavior"
     tools:context=".activity.LandingActivity"
     tools:showIn="@layout/activity_landing">

     <LinearLayout
         android:id="@+id/landing_content"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:orientation="vertical"/>

 </android.support.v4.widget.NestedScrollView>

</LinearLayout>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:orientation="vertical"
    tools:context=".activity.MainActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>

    <android.support.v4.widget.NestedScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:context=".activity.MainActivity"
        tools:showIn="@layout/activity_main">

        <LinearLayout
            android:id="@+id/main_content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"/>

    </android.support.v4.widget.NestedScrollView>

</LinearLayout>
```

activity_landing_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <ImageView
                android:id="@+id/landing_card_image"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="@dimen/image_margin"
                android:background="@mipmap/content"/>

            <TextView
                android:id="@+id/landing_card_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:padding="8dp"/>
        </LinearLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_article_template_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:card_view="http://schemas.android.com/apk/res-auto"
  android:id="@+id/landing_card"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:layout_marginLeft="12dp"
  android:layout_marginTop="6dp"
  android:layout_marginRight="12dp"
  android:layout_marginBottom="6dp"
  card_view:cardCornerRadius="4dp">

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
```

```
            <TextView
                android:id="@+id/article_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="bold"/>

            <TextView
                android:id="@+id/article_created_by"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="italic"/>

            <TextView
                android:id="@+id/article_description"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"/>

            <TextView
                android:id="@+id/article_url"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="italic"
                android:visibility="gone"/>
        </LinearLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_contact_info_template_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:id="@+id/contact_info_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="bold"/>
```

```
<TextView
    android:id="@+id/contact_info_email"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:textStyle="italic"/>

<TextView
    android:id="@+id/contact_info_description"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="8dp"/>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <LinearLayout
        android:id="@+id/contact_info_facebook"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:visibility="gone">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/facebook"
            android:layout_margin="@dimen/image_margin"/>
```

```
                <TextView
                    android:id="@+id/contact_info_facebook_text"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_vertical"/>
            </LinearLayout>

            <LinearLayout
                android:id="@+id/contact_info_twitter"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:visibility="gone">

                <ImageView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@mipmap/twitter"
                    android:layout_margin="@dimen/image_margin"/>

                <TextView
                    android:id="@+id/contact_info_twitter_text"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_vertical"/>
            </LinearLayout>
        </LinearLayout>
    </LinearLayout>
</RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_location_template_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:id="@+id/location_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="bold"/>
            <TextView
                android:id="@+id/location_address"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="8dp"
                android:textStyle="italic"/>

            <ImageView
                android:id="@+id/location_image"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:scaleType="centerInside"
                android:layout_gravity="center"
                android:layout_margin="@dimen/image_margin"
                android:background="@mipmap/map"
                android:visibility="gone"/>
        </LinearLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_profile_template_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:card_view="http://schemas.android.com/apk/res-auto"
   android:id="@+id/landing_card"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:layout_marginLeft="12dp"
```

```
android:layout_marginTop="6dp"
android:layout_marginRight="12dp"
android:layout_marginBottom="6dp"
card_view:cardCornerRadius="4dp">

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/profile_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="centerInside"
            android:layout_margin="@dimen/image_margin"/>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_weight="1">

            <TextView
                android:id="@+id/profile_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
            android:padding="8dp"/>

         <TextView
            android:id="@+id/profile_description"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="8dp"/>
      </LinearLayout>

    </LinearLayout>
  </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_web_template_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:card_view="http://schemas.android.com/apk/res-auto"
  android:id="@+id/landing_card"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:layout_marginLeft="12dp"
  android:layout_marginTop="6dp"
  android:layout_marginRight="12dp"
  android:layout_marginBottom="6dp"
  card_view:cardCornerRadius="4dp">

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```

```
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <ImageView
                android:id="@+id/web_image"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="@dimen/image_margin"
                android:background="@mipmap/internet"/>

            <TextView
                android:id="@+id/web_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:padding="8dp"/>
        </LinearLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_image_element_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/image_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="centerInside"
            android:layout_centerInParent="true"
            android:layout_margin="@dimen/image_margin"/>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_link_element_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@mipmap/internet"
                android:layout_margin="@dimen/image_margin"/>

            <TextView
                android:id="@+id/link_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"/>
        </LinearLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_text_element_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/landing_card"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="6dp"
    android:layout_marginRight="12dp"
    android:layout_marginBottom="6dp"
    card_view:cardCornerRadius="4dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/text_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:padding="8dp"/>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

activity_main_youtube_video_element_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
```

```
android:id="@+id/landing_card"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginLeft="12dp"
android:layout_marginTop="6dp"
android:layout_marginRight="12dp"
android:layout_marginBottom="6dp"
card_view:cardCornerRadius="4dp">

<RelativeLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content">

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="@dimen/image_margin"
      android:src="@mipmap/youtube"/>

    <TextView
      android:id="@+id/youtube_video_title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_vertical"/>
  </LinearLayout>
```

```
    </RelativeLayout>
</android.support.v7.widget.CardView>
```